Anfragesprachen für Internet-Informationssysteme

Inauguraldissertation
zur Erlangung des Grades Dr. phil.
im Promotionsfach Bibliothekswissenschaft

vorgelegt an der philosophischen Fakultät I
der Humboldt Universität zu Berlin

von
Diplom Informatiker
Josef Willenborg
aus Berlin

Tag der Disputation: 17. Juli 2001
Erstgutachter: Prof. Dr. Walther Umstätter
Zweitgutachter: Prof. Dr. Friedrich Braun
Kommissionsvorsitz: Prof. Dr. Rainer Kuhlen

Dekan der Philosophischen Fakultät I: Prof. Dr. Wilfried Nippel



Für Pucki und Benjamin

Danksagungen

Mein besonderer Dank gilt Prof. Umstätter. Er hat diese Arbeit ermöglicht und die entscheidenden Weichenstellungen und Hinweise gegeben.
Prof. Braun danke ich für die wertvollen Anregungen insbesondere auf dem Gebiet der Thesaurus- und Wörterbuchforschung. Weiterhin danke ich ihm die fruchtbaren Treffen in der Zeit des ATLAS-Projekts und danach.
Besonders danken möchte ich auch Benedikt Homann und Roman Czyborra für ihre technische Unterstützung und ihre unermüdliche Bereitschaft, Fragen zu beantworten. Weiterhin danke ich Vincent Winczewski für seine Hinweise insbesondere zu Suchmaschinen.
Mein Dank gilt weiterhin Prof. Konrad und Dr. Reiner, die mit ihren Dissertationen eine Grundlage für diese Arbeit geschaffen haben.
Allen meinen Diplomand/innen danke ich für ihre zahlreichen Anregungen. Insbesondere danke ich Haider Hammoudi, Frank Hartlep, I Eng Kho, René Schönfeldt, Wolfram Schneider, Dirk Schulz und Edgar Weitzel.

Berlin, im März 2001

Inhaltsverzeichnis:

1 Einleitung

1.1 Ausgangspunkt

Seit Entstehung des World Wide Web hat sich das Internet vom reinen Wissenschaftsnetz zum "Netz für jedermann" entwickelt. Fast eine halbe Milliarde Internet-Benutzer greifen weltweit auf mehrere Milliarden von Web-Dokumenten zu[1]. Jeden Tag werden fast 1 Milliarde Suchmaschinen-Anfragen gestartet. Für die Zukunft wird ein weiterer starker Anstieg dieser Zahlen erwartet. Im Internet sind zahlreiche neue Informationssysteme entstanden. Zu den traditionellen Informationssystemen zur Dokumentensuche sind neue Klassen von Informationssystemen hinzugekommen. Suchmaschinen unterschiedlichster Art stellen Dokumente im Volltext bereit, Systeme wie z.B. Oracle Intermedia bieten eine Suche in geschachtelten Dokumenten an, erste Systeme bieten Netzwerkoperatoren für link-strukturierte Dokumente an.
Gleichzeitig kann mit diesem quantitativen Wachstum die qualitative Entwicklung der Informationssysteme nicht mithalten. Der Dokumentbestand der Systeme entspricht nicht dem versprochenen Umfang und ist inaktuell. Die Systeme spezialisieren sich auf Dokumenttypen und erfassen die Dokumente unterschiedlich. Weiterhin überlappt der Bestand der Systeme z.T. sehr stark. Die Systeme sind zeitweise wegen Netz-, Hard- oder Softwareproblemen unerreichbar. Der Dokumentbestand ist mit den bereitgestellten Suchmitteln (Suche, Darstellung, Navigation) inhaltlich nicht mehr durchschaubar. Oft sind die Suchmittel unterschiedlich, obwohl sie dasselbe ausdrücken. Weiterhin werden Suchmittel nicht den neueren Dokumenttypen wie link-strukturierte und geschachtelte Dokumente angepaßt. Oft sind die Suchergebnisse zu unpräzise. Die Anfragemächtigkeit in den Systemen ist unterschiedlich. Die Systeme bieten keinen einheitlichen Zugriff auf Internet-Dokumente an. Beispielsweise muß für die Anfrage nach Dokumenten, die das Wort "Hund" oder "Katze" enthalten, mit der Suchmaschine Alta-Vista die Anfrage "Hund Katze" gestartet werden, währenddessen mit der Suchmaschine Google zwei verschiedene Anfragen "Hund" und "Katze" gestartet werden und dann beide Suchergebnisse per Hand vereinigt werden müssen. Ein systematischer Vergleich der Anfragemöglichkeiten von Internet-Informationssystemen existiert bisher nicht.
Die wissensbasierte Suche mit Hilfe von Thesauri, Terminologien etc. wird entweder gar nicht oder nur unzureichend unterstützt. Inhaltliche Indexierungen können inaktuell und inkonsistent werden. Weiterhin indexiert der Internet-Nutzer bzw. -Autor Internet-Dokumente ohne Regeln und ohne feste Bindung an eine Wissensbasis u.U. bewußt fehlerhaft. Dadurch wird die Indexierungsqualität i.A. stark verschlechtert. Eine Pflege der Wissensbasis im Team von Indexierern, Terminologen und Suchenden, die ja durch die neuen Kommunikationstechniken möglich wäre, wird bisher nicht durchgeführt.
Die bittere Folge für den Benutzer: er hat zwar im Prinzip große Zugriffsmöglichkeiten (in kurzer Zeit, von jedem Ort, in vielen Formen, mit vielfältigen Anfragemöglichkeiten), kann sie jedoch nicht adäquat nutzen und findet nicht das, was er braucht.

1.2 Problemstellung der Arbeit

Es ist ein offenes Problem, einen universellen Zugang zu Internet-Informationssystemen zu schaffen, der die Anfragemächtigkeit der einzelnen Systeme ausreichend berücksichtigt.
Voraussetzung zur Entwicklung eines solchen Zugangs ist, daß vorhandene Anfragemöglichkeiten in den vorhandenen Systemen systematisch untersucht werden. Dabei müssen sowohl traditionelle Systeme zur Dokumentensuche als auch neuere Systemklassen wie z.B. Systeme

[1] siehe Kap. 2.1.1

zur Suche in link-strukturierten Dokumenten, Systeme zur Suche in geschachtelten Dokumenten, Faktensuchsysteme, Systeme zur Suche nach terminologischen Einträgen und intelligente Agenten berücksichtigt werden. Damit wichtige Anfragemöglichkeiten nicht übersehen werden, müssen für jedes System Anfragebeispiele in jeweils systemspezifischer Syntax vorgestellt werden. Anhand der gewonnenen Ergebnisse kann dann ein systematischer Vergleich anhand wichtiger Kriterien wie Feld/Formatstruktur, Bestand, Suchmittel, Suchergebnisse und Leistungsgrenzen durchgeführt werden.
Der zweite Schritt zur Lösung des Problems besteht darin, Anfragesprachen mit spezifizierter Syntax und Semantik für die Systemklassen (hier: Suche in einfach strukturierten Dokumenten, Suche in link-strukturierten Dokumenten, Suche in geschachtelten Dokumenten) aufzubauen und zu einer universellen Anfragesprache zusammenzuführen. Diese fungiert als Zwischensprache in einem universellen Internet-Suchsystem.
Der dritte Lösungsschritt besteht in dem Entwurf einer Systemarchitektur für solch ein universelles Internet-Informationssystem, das die im zweiten Schritt entwickelte Anfragesprache verwendet.
Bei der Entwicklung eines universellen Zugangs für Internet-Informationssysteme ergibt sich zusätzlich zu den eher formalen und technischen Fragestellungen das Problem des inhaltlichen Zugriffs auf Internet-Dokumente. Dafür wird in dieser Arbeit eine Lösung dreier Teilprobleme herbeigeführt:
1. Momentan unterstützen terminologiebasierte Informationssysteme neuere Techniken aus der Hypertextforschung noch nicht ausreichend. Durch den Einsatz von Such- Browsing- und Darstellungstechniken aus dem Hypertextbereich kann das terminologiebasierte Retrieval verbessert werden.
2. Die Qualität des terminologiebasierten Retrievals kann durch die Verbesserung der Konsistenz von Terminologien erhöht werden.
3. Durch die Entwicklung eines Kommunikationsmodells für die terminologische Indexierung von Internet-Dokumenten, das den neuen Komunikationstechniken Rechnung trägt, wird die Qualität der Indexierung erhöht.

1.3 Aufbau der Arbeit

In Kapitel 2 werden zunächst Grundlagen für diese Arbeit geschaffen. Es wird der Stand der Entwicklung in den Bereichen Internet, Hypertext, Graphentheorie, Logik und Anfragesprachen dargestellt.
In Kapitel 3 wird eine vergleichende Untersuchung der Anfragemöglichkeiten von Internet-Informationssystemen durchgeführt. Systeme werden in Systemklassen eingeteilt, die sich auf die Art der Suche (Suche in einfach strukturierten Dokumenten, Suche in link-strukturierten Dokumenten etc.) beziehen. Im letzten Teilkapitel wird eine Zusammenfassung des Vergleichs durchgeführt.
In Kapitel 4 werden aufbauend auf den Ergebnissen von Kapitel 3 Anfragesprachen für drei Systemklassen von Internet-Informationssystemen entwickelt und zu einer universellen Anfragesprache zusammengeführt. Für jede Anfragesprache werden Syntax und Semantik definiert und Anfragebeispiele gegeben.
In Kapitel 5 wird die Architektur eines Internet-Informationssystems entworfen, das die entwickelte Anfragesprache unterstützt.
In Kapitel 6 wird das terminologiebasierte Information Retrieval in den Kontext der Hypertextforschung gebracht. In den beiden ersten Teilkapiteln werden grundlegende Definitionen gegeben und Verfahren des terminologiebasierten Information Retrieval vorgestellt.. Im dritten Teilkapitel werden zur Konsistenzverbesserung von Terminologien formale Eigenschaften

von Relationen eingeführt. Im letzten Teilkapitel wird ein Kommunikationsmodell für das terminologiebasierte Information Retrieval entworfen.
In Kapitel 7 und 8 werden die Ergebnisse der Arbeit diskutiert und schließlich in Thesen vorgestellt. Weiterhin werden zukünftige Entwicklungsmöglichkeiten aufgezeigt.
In Kapitel 9 wird ein Ausblick gegeben.
Im Anhang wird das in der Arbeit verwendete Material (Literatur, URL von Personen, Organisationen und Produkten, Normen und Symbole) aufgeführt. Weiterhin wird ein eigenentwickelter Prototyp vorgestellt, der einen Teilausschnitt einer Anfragesprache realisiert.

Vorarbeiten für diese Arbeit wurden im interdisziplinären Forschungsprojekt ATLAS an der TU Berlin (ATLAS (1993)) durchgeführt. Dort wurde zusammen mit dem Institut für Linguistik (Prof. Braun) ein Prototyp zur Thesaurus- und Wörterbuchpflege entwickelt (Willenborg (1991a), Willenborg (1991b), Willenborg (1994a)) und ein Modell für eine qualitative Indexierung von Dokumenten entworfen. Weiterhin wurden erste theoretische Konzepte für das terminologiebasierte Information Retrieval (Willenborg (1994b)) und die konsistente Zusammenführung von Terminologien entwickelt. Sprachentwürfe einer Thesaurusanfragesprache wurden geschaffen.

2 Grundlagen

2.1 Internet

2.1.1 Entstehung und Wachstum

Das Internet entstand Anfang der 70er Jahre als eine Verbindung des ARPAnet des amerikanischen Verteidigungsministeriums mit Radio- und Satellitennetzwerken. Dieses Netz basierte auf dem Network Control Protocol (NCP). 1981 wurde die Einbindung weiterer Netzwerke durch die Entwicklung des standardisierten Protokolls TCP/IP (RFC 791, RFC 793) ermöglicht. Zu diesem Zeitpunkt waren 200 Rechner an das Internet angeschlossen. Durch eine Initiative der US National Science Foundation (NFS) wurde die Zahl der weltweit angeschlossenen Rechner bis 1989 auf 150000 gesteigert. Anfang 2000 bestand das Netz durch den Boom, den das World Wide Web ausgelöst hatte, aus 72 Millionen per DNS an das Internet angeschlossenen Computern.
Das Internet wird durch die folgenden charakteristischen Eigenschaften zusammengehalten:
1. Es ist standardisiert: TCP/IP ist der Standard für den Zusammenschluß von Computer-(netzwerken) weltweit geblieben. Zahlreiche weitere Protokolle, Computernetzwerke und Client-Server-Anwendungen können gemeinsam benutzt werden.
2. Es ist benutzerfreundlich: Anwender entwickeln es gemeinsam weiter. Weiterhin haben sich zahlreiche Diskussions- und Arbeitsgruppen gebildet.
3. Es ist effizient: Der Austausch von Daten geschieht über Netze.
4. Es ist dezentral.
Folgende Zahlen belegen den Stellenwert des Internet:

	Zahl	Quelle	Stand
Web-Dokumente davon indiziert	4 Milliarden 1,25 Milliarden (Google) 0,57 Milliarden (Fast)	Cyveillance (2001) SearchEngineWatch (2001)	Februar 2001 November 2000
Internet-Nutzer	333 Millionen	Nua (2000a)	Juni 2000
Internet-Suchen	50 Millionen/Tag (Altavista) 40 Millionen/Tag (Google)	SearchEngineWatch (2001)	November 2000
E-Mails	5 Milliarden/Tag	Computer Zeitung vom 2.11.2000	November 2000
E-Mail-Listen	90.000 (Liszt)	Liszt (2000)	August 2000
Newsgruppen	35.000 (Dejanews)	DejaNews (2000b)	August 2000
Internet-Rechner	72,4 Millionen	ISC (2000)	Januar 2000
HTTP-Server	18,2 Millionen	Netcraft (2000)	Juli 2000
Internet Service Provider	8.000	Mids (2000)	Dezember 1999
Web-Umsatz	3,9-36 Milliarden $ 1999 (Schätzung verschiedener Unternehmen)	Nua (2000b)	August 2000

Tabelle 1: Größe des Internet

2.1.2 Informationssysteme

Shannon und Weaver definieren Information als den Logarithmus der Anzahl der Auswahlmöglichkeiten von Zeichen (Shannon, Weaver (1963)). Der mittlere Informationsgehalt einer Nachricht wird mit $H = - \Sigma P_i * \log_2 P_i$ gleichgesetzt, wobei P_i die Wahrscheinlichkeit bedeutet, mit der die jeweiligen Zeichen dieser Nachricht eintreffen. Wenn beispielsweise das Alphabet A = {a, b, c, d, e, f, g, h} gegeben ist und jedes dieser Zeichen mit der Wahrscheinlichkeit von 1/8 eintrifft, dann hat eine Nachricht mit der Länge von 8 Zeichen den mittleren Informationsgehalt von:

$$H = -((1/8 * \log_2 1/8) * 8) = -((1/8 * -3) * 8) = 3$$

Nach Salton (1983) S. 7 ist der Begriff Informationssystem Oberbegriff für Managementinformationssystem, Datenbankmanagementsystem, Entscheidungsunterstützungssystem, Frage-Antwortsystem und Information Retrieval System. Bei Reiner (1991) S. 35 ist der Begriff Informationssystem Oberbegriff für Dokumenten-, Fakten -und Erklärungssuchsystem.

Ein Informationssystem ist ein System, das der Speicherung und Wiedergewinnung von Information dient. Information liegt in informetrischen Einheiten vor. Ein besonderer Typ informetrischer Einheiten sind die Dokumente.
"Dokumente sind als Oberbegriff verschiedener Dokumenttypen (Akten, Bilder, Bücher, Briefe oder Tonaufzeichnungen) handhabbare informetrische Einheiten, die sich auf verschiedenen Informationsträgern befinden können und damit sehr unterschiedlichen Umfang und variable Gestalt zeigen." (Ewert, Umstätter (1997) S.164).

Ein weiterer Dokumenttyp, der in dieser Arbeit eine besondere Rolle spielt, sind die elektronischen Dokumente. Diese lassen sich folgendendermaßen einteilen:

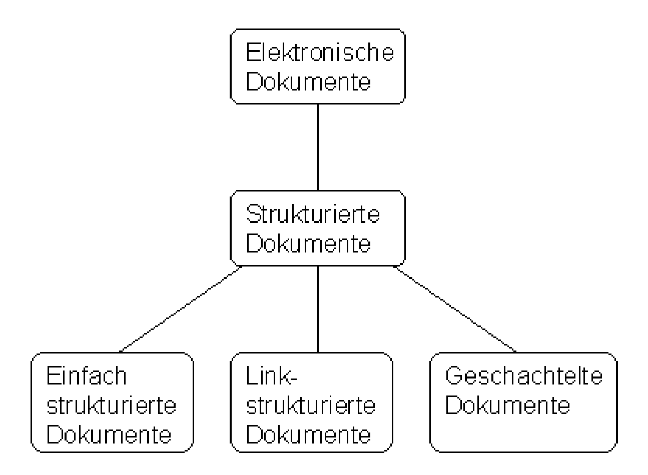


Abbildung 1: Elektronische Dokumente

Einfach strukturierte Dokumente bestehen aus formalen und inhaltlichen Deskriptoren. Beispiele sind Volltextdokumente, unstrukturierte Multimediadokumente, Personeneinträge, Rechnereinträge, Netzwerkeinträge, Dateien, E-Mails, News-Artikel, Gopher-Menüs, Dokumenthinweise etc. In link-strukturierten Dokumenten werden Dokumente durch Beziehungen miteinander verbunden. In geschachtelten Dokumenten können Teilstrukturen der Dokumente aufgebaut werden.
Weiterhin können Teilelementen eines Dokuments Formatierungen wie z.B. Größe, Farbe, Lautstärke, Schriftart etc. zugeordnet werden.

SGML- bzw. XML-Dokumenten sind sowohl link-strukturiert als auch geschachtelt. Die Schachtelung wird durch eine DTD (Document Type Definition) definiert. Links können innerhalb und außerhalb eines Dokuments definiert werden. Formatierungen werden über eine Abbildung der DTD-Elemente auf Formatelemente definiert. Beispiele dafür sind CSS (Bos, Jacobs et.al. (1998)) und XSL (Adler, Berglund et.al. (2000)).
In HTML-Dokumenten werden Strukturierung und Formatierung nicht voneinander getrennt.

Der Dokumentenbestand, im weiteren Bestand genannt, ist die Dokumentenmenge in einem Informationssystem.

Eine besondere Form von Informationssystemen sind die wissensbasierten Systeme.
„Alle Formen von Nachrichten, d.h. Information, Rauschen, Redundanz und Wissen, sind in Bit meßbar. (...) Wissen ist eine besonders effiziente Informationskompression. Es ist als eine begründete Information zu verstehen. (...) Wissen ist vorausschauend und probabilistisch. Sein Wert ist hoch, wenn die Voraussagen (...) korrekt sind, im Gegensatz zur Information, die possibilistisch ist, da die unwahrscheinlichsten Zeichen den höchsten Informationsgehalt tragen“ (Umstätter (1998) S. 221-224).
Wissen läßt sich in Wissensarten einteilen: sicheres Wissen, unsicheres Wissen, unvollständiges Wissen, vages Wissen etc.
Wissensbasierte Systeme bestehen aus den Komponenten Wissensbasis (Fakten, Regeln), Inferenzsystem und Benutzerschnittstelle. Wissensbasierte Systeme bieten die Möglichkeit mit Hilfe logischer Regeln (Induktion, Deduktion etc.) aus vorhandenem Wissen weiteres Wissen (Fakten, Regeln) abzuleiten.
Die Suche mit einem wissensbasierten System kann das Retrievalergebnis verbessern, weil das Wissen des Systems nutzbar ist. So kann i.A. die Effizienz der Suche nach Dokumenten mit einem Thesaurus erhöht werden.

Traditionell häufig verwendete Maße zur Bestimmung der Effizienz von Informationssystemen sind Recall-Ratio, Precision und Noise:
Recall-Ratio ist die Anzahl der gefundenen relevanten Dokumente im Verhältnis zu allen relevanten Dokumenten in der zugrundeliegenden Dokumentenbasis bezüglich einer Anfrage.
Precision ist die Anzahl der gefundenen relevanten Dokumente im Verhältnis zu allen gefundenen Dokumenten bezüglich einer Anfrage.
Noise ist die Anzahl der gefundenen nicht relevanten Dokumente im Verhältnis zu allen nicht relevanten Dokumenten in der Dokumentenbasis bezüglich einer Anfrage.

2.1.3 Protokolle im Client-Server-Modell

Viele Informations- und Kommunikationsdienste werden im Client-Server-Modell realisiert.
"*There is a set of server processes, each acting as a resource manager for a collection of resources of a given type, and a collection of client processes, each performing a task that requires access to some shared hardware and software resources. Resource managers may themselves need to access shared resources managed by another process, so some processes are both client and server processes.*" (Coulouris, Dollimore, Kindberg (1994) S.11-12)

Die Kommunikation im Client-Server-Modell wird i.A. anhand eines Schichtenmodells beschrieben. Man unterscheidet die 7 Schichten: Physikalische Schicht, Bitübertragungsschicht, Netzwerkschicht, Transportschicht, Sitzungsschicht, Präsentationsschicht und Anwendungsschicht. Jede Schicht kann die Dienste der jeweils unter ihr liegenden Schichten in Anspruch

nehmen, ohne deren Funktionsweise zu kennen. Jede Schicht kommuniziert mit einer Schicht der gleichen Ebene über ein festgelegtes Protokoll[2].

Man unterscheidet die verbindungsorientierten und die verbindungslosen Kommunikationsprotokolle.
"Damit der vergleichsweise hohe Aufwand der verbindungsorientierten Protokolle, wie z.B. OSI oder TCP/IP, vermieden wird, basiert das Client-Server-Modell meistens auf einem einfachen, verbindungslosen Anfrage/Antwortprotokoll. (...) Ein Client sendet eine Anfrage und erhält eine Antwort." (Tanenbaum (1995) S. 494)
Ein Anfrage- bzw. Antwortprotokoll legt Regeln fest, die bestimmen, ob Anfragen und Antworten syntaktisch korrekt gebildet sind und somit gesendet bzw. empfangen werden können. Der Anfrage-Client kennt von seinem Antwort-Server nur das Protokoll, nach dem die Kommunikation abläuft.

Ein verläßliches Protokoll ist ein Protokoll, welches
1. sicherstellt, daß gesendete Daten beim Kommunikationspartner ankommen. Falls Daten beim Kommunikationspartner zunächst nicht ankommen, werden sie erneut bis zum erfolgreichen Empfang dorthin gesendet.
2. die Berücksichtigung der Reihenfolge der gesendeten Daten gewährleistet.
Um Anfrage- bzw. Antwortprotokolle verläßlich zu machen, wird meist die Netzwerk- und Transport-Schicht verwendet.

Das Internet wird durch ein 4-Schichten Modell gebildet: physikalische und Bitübertragungsschicht, Netzwerkschicht (RFC 791), Transportschicht (RFC 793) und Applikationsschicht.

Man unterscheidet folgende Kommunikationsprotokolle (Applikationsschicht) von Internet-Informationssystemen (Server, Clients):

Dokumenttyp	Protokoll	Server	Client
Person, Gruppe	Finger User Information Protocol nach RFC 742 und RFC 1288	Finger	Finger, Netfind
	Nicname Whois nach RFC 954	Whois	Telnet, Whois
	Directory Access Protocol (DAS) / X.500 nach RFC 1308, RFC 1309, RFC 1292 bzw. ISO 9594	Directory System Agents	Directory User Agents
Rechner, Netzwerk	Domain Name System (DNS) nach RFC 1034	Domain Name Service	Domain Name Resolver
	Nicname Whois nach RFC 954	Whois-Service	Telnet, Whois, ...

[2] *"Grundsätzlich ist ein Protokoll eine Übereinkunft darüber, wie Kommunikation vonstatten zu gehen hat."* (Tanenbaum (1995) S. 486)

Datei	Telnet Protocol nach RFC 854	Telnet	Telnet, HyTelnet
	File Transfer Protocol (FTP) nach RFC 959	FTP	FTP, Netscape,
	Prospero Protocol	Prospero	Archie, Prospero
	XDR nach RFC 1014 Remote Procedure Call (RPC) nach RFC 1057	Alex	Alex
E-Mail	Simple Mail Transfer Protocol (SMTP) nach RFC 821, 822	Sendmail	Mail, Elm, Netscape
	Message Handling System (MHS) nach X.400, ISO 10021	Message Transfer Agent	User Agents
Artikel	Network News Transfer Protocol (NNTP) nach RFC 977	Usenet News	Rn, Trn, Netnews, Tin
Menü	Gopher Protocol nach RFC 1436	Gopher	Xgopher, Netscape
Dokument hinweis	Information Retrieval Service and Protocol Standard (Z39.50)	Wide Area Information System (WAIS), Z39.50	waissearch, freeWAIS, freeWAIS-sf, Z39.50
Hypertext-Dokument	Hypertext Transfer Protocol (HTTP) nach RFC 1945	HTTP	Netscape, Internet Explorer
	Hyper-G-Protokoll (HG-CSP) nach Kappe, Pani (1996)	Hyper-G	Harmony, Amadeus
Fakten	ODBC, JDBC	Protokolle der Datenbanksysteme	ODBC, JDBC

Tabelle 2: Protokolle von Internet-Informationssystemen

Um die Herkunft der Daten aus einer verläßlichen Quelle zu gewährleisten, werden in den Protokollschichten Mechanismen zur Authentifizierung vorgehalten (Stallings (1995), SHTTP (1996), SSL (1996)).

2.1.4 Zeichensätze

Ein Zeichensatz ist eine zum Schreiben benutzte Menge von graphischen Symbolen (vgl. O'Donell (1994)). In Internet-Informationssystemen wird überwiegend der 8-Bit Zeichensatz nach ISO 8859 verwendet. Ausnahmen sind Mail, FTP, News und bestimmte Dateisysteme, bei denen z.T. noch der 7-Bit Zeichensatz nach ISO 646 (US-ASCII) benutzt wird.
Damit 7- und 8-Bit-basierte Systeme miteinander kommunizieren können, existieren Konvertierungswerkzeuge. Für Mail ist das zum Beispiel MIME (RFC 1521, RFC 1522), mit dem automatisch eine Konvertierung in ein anderes Format durchgeführt werden kann.

Vorteile der ISO 8859 Zeichensätze sind:
1. statt 128 Zeichen können 256 Zeichen innerhalb eines Zeichensatzes verwendet werden
2. gegenüber dem 16-Bit Zeichensatz wird eine Speicherplatzersparnis von 50% erreicht
3. zwischen den regional unterschiedlichen Zeichensätzen (im Moment 10) kann relativ einfach gewechselt werden

Die Zukunft gehört dem Zeichensatz Unicode (Unicode (2001), ISO 10646, Czyborra (2001)) mit den Kodierungen wie z.B. UCS-2 (16 Bit) und UTF-8 (Multibyte). Der Hauptvorteil von Unicode ist, daß 65536 Zeichen innerhalb eines Zeichensatzes dargestellt werden können, so daß alle international häufig verwendeten Zeichen codiert werden können. Die vollständige Umstellung der Systeme auf Unicode wird jedoch noch einige Zeit in Anspruch nehmen. Das erste auf Unicode basierende Betriebssystem ist Windows NT. Die objektorientierte Programmiersprache Java (Flanagan (1996), Middendorf, Singer, Strobel (1996)) ist eine der ersten Programmiersprachen, die Unicode intern verwendet.

2.1.5 Reguläre Ausdrücke

Eine mächtige Möglichkeit der Volltextsuche bietet die Suche mit sogenannten regulären Ausdrücken. Dabei werden Metazeichen und Fluchtzeichen verwendet. Ein Metazeichen ist ein Zeichen des Alphabets, das eine eine Bedeutung über das Zeichen hinaus hat. Beispiele von Metazeichen in regulären Ausdrücken sind die Zeichen "? * . \ { [ ] $ |". Ein Fluchtzeichen ist ein Zeichen des Alphabets, das die Metabedeutung des direkt nachfolgenden Zeichens aufhebt. Beispiel ist das Zeichen "\". Gulbins, Obermayr (1995) geben einen Überblick über Flucht- und Metazeichen in regulären Ausdrücken und ihre Bedeutung. Reguläre Ausdrücke können z.B. in Unix-Betriebsystemkommandos oder in dem Textverarbeitungssystem Microsoft Word (2000) verwendet werden.

2.2 Hypertext, Hypermedia

2.2.1 Was ist Hypertext, Hypermedia

Die Grundidee bei Hypertext besteht darin, Information nicht sequentiell, sondern in einem vernetzten Gewebe (Netzwerk, Web, Net) zu präsentieren. Ein Hypertext besteht aus einer Menge von Dokumenten und ihren Beziehungen untereinander (Nielsen (1996)).
Der Begriff Hypermedium weist zusätzlich auf den multimedialen Charakter der Dokumente hin. Der Begriff Hypertext wird im Folgenden als Synonym für Hypermedium verwendet.

Hypertext-Dokument (Synonyme: Knoten, Frames, Karten, Rahmen, Fenster etc.):
Hypertext-Dokumente sind Dokumente, die über Hypertextbeziehungen miteinander verbunden sind. Sie können wieder aus Hypertextdokumenten bestehen (geschachteltes Hypertext-Dokument).

Hypertextbeziehung (Synonyme: Verknüpfung, Link, Verbindung etc.):
"Eine Hypertextverbindung verbindet zwei Knoten und zeigt normalerweise von einem Knoten (den Ausgangsknoten) zu einem anderen Knoten (dem Zielknoten). Hypertextverbindungen werden häufiger mit spezifischen Teilen der Knoten und nicht mit dem gesamten Knoten in Zusammenhang gebracht." (Nielsen (1996) S. 2)

Wenn Leser sich durch das Netz bewegen, bezeichnet man dies als Blättern (engl. browsing) oder Navigieren.

2.2.2 Geschichtliche Entwicklung

Die Entwicklung erster theoretischer Konzepte für Hypertextsysteme liegt ca. 50 Jahre zurück. Pionier ist Vanevar Bush (Bush (1945)). Sein System MEMEX basiert auf der Mikrofilm- und Magnetbandtechnik, die es dem Betrachter von Wissensbeständen erlaubt, selbst- oder fremdgelegte Spuren zu benennen und zu verfolgen, in den Beständen zu wandern und zu springen, assoziativ zu indexieren, oder globale Bestände mit dem "Wissen eines ganzen Volkes" zu erzeugen. Bush erkannte, daß es wichtig ist, die unterschiedlichen Wahrnehmungsorgane des Menschen wie Tast-, Hör-, Sprach- und Sehsinn auch mit den erzeugten Medien, den Multimedien, anzusprechen.

Der Begriff Hypertext wurde zum ersten Mal von Ted Nelson im Jahre 1960 im Zusammenhang mit seiner Idee globaler Bibliotheken (*docuverses)* verwendet.

Es hat lange gedauert, bis diese Ideen technisch effektiv realisiert werden konnten.

Das erste auf einem Computer lauffähige Hypertextsystem (Maus, Fensteroberfläche) war das von Douglas Engelbart 1968 vorgestellte System AUGMENT (Engelbart (1984)), das vorhandene Wissensbestände um individuelles und organisatorisches Wissen anreichert.

Eines der am weitesten verbreiteten Hypertextsysteme ist das beim Kauf von MacIntosh-Computern kostenlos mitgelieferte System HYPERCARD (Hypercard (1997)). Es bietet objektorientierte Programmiermöglichkeiten mit der Sprache Hypertalk.

Weitere Hypertextautorensysteme sind bespielsweise GUIDE (Brown (1989)), XANADU (Nelson (1993)) und HYPERTIES (Kreitzberg, Shneiderman (1988)).

Heute existiert eine Vielzahl von Hypertextsystemen in unterschiedlichen Anwendungsgebieten wie Textverarbeitung, Information Retrieval, computergestützter Unterricht, Lexikon- und Thesauruspflege, Kommunikation. Hypertext-Hilfesysteme gibt es zum Beispiel für Word-Perfect, Microsoft Word, Microsoft Windows, OS-2 und SunOS.

Für den Zugriff auf Hypertextbestände im World Wide Web werden Benutzeroberflächen wie zum Beispiel Netscape (Netscape (2001)) und Microsoft Internet Explorer (Explorer (2001)) verwendet. Diese arbeiten mit dem Protokoll HTTP (HTTP (1996)), das die Internetprotokolle FTP, NETNEWS, MAIL, GOPHER, und WAIS zusammenfaßt und um weitere Funktionen für Hypertext erweitert (Berners-Lee (1993)). Hypertextbestände werden mit der international standardisierten Dokumentenbeschreibungssprache HTML (Ragett (1997)) und XML (Goldfarb, Prescod (2001)) erfaßt.

Die Benutzeroberfläche Harmony (Dalitz, Heyer (1995), Maurer (1996)) verwendet als Protokoll Hyper-G (Kappe, Pani (1996)). Hypertextbestände werden mit der Dokumentenbeschreibungssprache HTF (Kappe (1996)) erfaßt.

Seit 1996 werden zunehmend Suchmaschinen eingesetzt. Suchmaschinen realisieren die Suche über einen großen Teil des weltweit öffentlich zugänglichen Dokumentbestandes (vor allem HTML).

In letzter Zeit ist eine Entwicklung hin zur Unterstützung von Multimedia bzw. Hypermedia zu erkennen. Weiterhin werden geschachtelte Dokumente abgebildet.

1945	Vannevar Bush (Bush (1945)) in Atlantic Monthly: As we may think. Schlagworte: MEMEX, microfilm, card, web of trails, association.
1960/65	Ted Nelson mit dem Projekt XANADU. Schlagworte: Hypertext, docuverses, hot links, hypermedia information server
1967/68	erste lauffähige Hypertextsysteme: Brown University: Hypertext Editing System; FRESS; (beide IBM/360 mit 128KB RAM)
1968	Douglas Engelbart: Stanford: AUGMENT (Graphik, Maus, Fensteroberfläche)
1978	erstes Hypermediasystem: Andrew Lippman, MIT: ASPEN MOVIE MAP: Autofahrt durch die Stadt Aspen mit Hilfe von Fotographien der Straßen, <= 2 Bilder/sec = 110 km/h; technisch 30 Bilder/sec möglich, Bildannotationen
1985	Symbolics Dokumentation als Hypertext (10000 Dokumente, 23000 Beziehungen, 10MB, nicht sehr häufig eingesetzt)
1986	erstes auf einem PC lauffähiges System: GUIDE, University of Kent, Brown 1988 (Mac, PC) INTERMEDIA, Brown Univ., A. van Dam: Lernen englischer Literatur(Mac, SunOS) HyperTIES, Univ. of Maryland, Kreitzberg, Shneiderman (1988): SunOS, DOS
1987	HYPERCARD von Bill Atkinson (Hypercard (1997), Mac, wird beim Kauf eines MacIntosh Computers kostenlos mitgeliefert) erste Hypertext Konferenz: ACM Hypertext 1987 in Chapel Hill, North Carolina mit 250 Teilnehmern
1989	HAUPT-AUTOREN-SYSTEM: (PC), TOOLBOOK (PC) zweite Hypertext Konferenz: ACM Hypertext 1989 in Pittsburgh, 600 Teilnehmer erste wissenschaftliche Zeitschrift: Hypermedia
1990	erste europäische Konferenz: ECHT 1990 in Versailles
1992	Gründung ACM SIGLINK: 3 Rundschreiben im Jahr
1993	Web-Standards, CERN, Genf: URL, HTTP, HTML 1.0 und 2.0 erste Web-Server: CERN, NSCA erste Web-Benutzeroberflächen: MOSAIC 1.01, September: MOSAIC 2.0 (Unix, Mac, PC)
1994	erste Konferenz WWW 1994 (Mai) in Genf, 380 Teilnehmer zweite europäische Konferenz: ECHT 1994 (September) in Edinburgh zweite Konferenz WWW 1994 (17-20. Oktober) in Chicago, 1300 Teilnehmer, davon 650 aus der Industrie HTML 3.0 erste Hyper-G-Systeme Web-Benutzeroberflächen: Mosaic 2.5, Netscape, Panorama, WebExplorer von IBM, HotMetal-Editor, Lynx 2.3, Harmony
1995	dritte Konferenz WWW 1995 (10-14. April) in Darmstadt; ACM Hypertext 1995 in Washington; IBM- und Microsoft-Betriebssysteme mit direktem Internetanschluß über Modem oder ISDN erste Suchmachine: Harvest Web-Server Netscape, Apache, Harvest, ... Breiter Einsatz von Authentifizierungs- und Verschlüsselungssoftware Kommerzialisierung des Internet, Online-Bezahlungen VRML (Version 1.0)

1996	WWW-Server von Microsoft, Oracle, ... erster WWW-Server von Netscape mit SSL, erste Zertifizierungsstellen CSS1 (Cascading Style sheet, level 1: simple visual formatting model) zahlreiche WWW-Benutzeroberflächen: Internet Explorer, ... Suchmaschinen: AltaVista, Lycos, Webcrawler, ... Web-Datenbankkopplungen, erste "intelligente Agenten" Durchbruch der Programmiersprache Java
1997	HTML 4.0 Java Development Kit (Version 1.1) VRML (Version 97)
1998	Multimedialisierung: Quicktime, SMIL (Version 1.0), MPEG (Version 4), Streaming-Systeme, ... Textverarbeitungssysteme werden hypertextfähig, zahlreiche Hypertexteditoren PDF (Acrobat Version 1 – 4) CSS2 (Cascading Style sheet, level 2: media-specific style sheets) XML (Version 1.0), Document Object Model (Version 1.0), XQL
1999	Java Development Kit (Version 2) Handhelds mit Java-fähiger Web-Benutzeroberflächen XSL, Xpath, Xlink, Xpointer, XML-Namespaces
2000	XHTML (HTML 4 ausgedrückt in XML) MPEG (Version 7: Anfragen an geschachtelte Dokumente, XML-basiert)

Tabelle 3: Entwicklung von Hypertextsystemen

2.2.3 Was bringt Hypertext (mit sich)

Hypertext besitzt im Vergleich zu traditionellen linearen Textbeständen
1. Eigenschaften, die einen Mehrwert und
2. Eigenschaften, die einen Minderwert darstellen.

Eigenschaften von Hypertexten, die einen Mehrwert darstellen:

Elektronische Verarbeitungsmöglichkeiten

- Hypertext ist effektiv speicher- und zugreifbar:
  Beispiel: Größe, Preis, und Zugriffszeit eines Buches (mit 300 Seiten und 100 Bildern):
  Größe des Buches: ca. 10 MByte bzw. 5 cm^3 (bei 5 KByte/Seite, 100 KByte/Bild)
  Speicherpreis (ohne Wartungskosten): ca. 0,15 DM (Stand Januar 2001)
  Zugriffszeit: ca. 40 Sekunden (bei 2 MBit/Sekunde)
- Hypertext ist aus großer Entfernung verfügbar.
- Physikalisch einfach vorhandene Hypertexte können von vielen gleichzeitig benutzt werden, Kopien von Hypermediabeständen sind einfach zu erstellen.
- Hypertext kann modifiziert werden

Darstellung/Repräsentation

- Neue Medien: Ton, Bild, Dia, Film, Video, mehrdimensionale Darstellungen, Farbe, Geruch etc.
- Aufnahme- und Wiedergabemöglichkeiten für neue Medien
- Darstellungsformen der Dokumentenbestände: graphisch, hierarchisch etc.

- Teilbestände sind aus- und einblendbar. Dadurch entstehen zur Bearbeitungszeit dynamische Zusammenhänge (z.B. je nach Nutzergruppe unterschiedlich etc.)
- Der Effekt, daß das referenzierte Objekt vorhanden ist, schafft Raum für das eigentliche oder Darzustellende.
- Das Sammeln und Gegenüberstellen von Information ist einfach: virtuelle Ausstellungen, Sammlungen etc.
- Mit Anfragekommandos, Darstellungs- und Navigationsmitteln kann Information effizient selektiert werden.

Deskriptive Elemente
- Markierungen (Schriftgröße, -art, -typ, Bezeichnung von Textteilen etc.)
- Verweise: Zitate, Anmerkungen, Fußnoten, Literaturangaben, siehe-Anweisungen, Abkürzungen, Invertierte Dateien (Autoren, Abbildungen, Inhaltsverzeichnisse, Glossare etc.) etc.
- Deskriptoren: (formaler Art: Autor, Verlag, ..., inhaltlicher Art: Schlagwörter, inhaltliche Deskriptoren etc.)

(2) Eigenschaften von Hypertexten, die einen Minderwert darstellen:

Benutzung

- Texte können ca. 30% schneller auf dem Papier gelesen werden (vgl. Nielsen (1996) S. 281ff)
- Benutzer müssen sich an das neue Medium, an neue Werkzeuge gewöhnen. Anstatt zu lesen, schreiben und rechnen lernt der (Grund)-Schüler nun zusätzlich: fotografieren, visualisieren, gestalten, entwerfen, vernetzen, Geräusche erzeugen, Filme drehen, Geschmacks- und Geruchseindrücke vermitteln etc.
- Ohne Redaktionskonzept wird die Veröffentlichungsschwelle leicht überschritten. Es entsteht eine Menge an Geschwätz, dessen Speicherbarkeit und Zugriff problematisch ist.
- Es ist ein beträchtlicher Aufwand für den Aufbau des Hypertextbestandes zu leisten: 1. Vorhandene sequentielle Bestände in den aufzubauenden Hypertextbestand integrieren und modularisieren 2. Multimedialisieren 3. Multilingualisieren und 4. Vernetzen.
- Das Gesamtwerk eines Autor oder einer Autorengruppe ist durch die Verweisstruktur (z.B auf Dokumente anderer Autoren, die nicht zu erkennen sind) oft nicht mehr abzugrenzen.
- Die Größe der Hypertextdokumente nimmt im Vergleich zu traditionellen Dokumenten ab.
- Das Gefühl, durch einen Verweis einen eigenständigen Inhalt ausgedrückt zu haben, drängt die Kunst der Erzeugung einer textinternen Verweisstruktur zurück und fördert die Tendenz, zu referenzieren. Durch Referenzierungswut können inhaltsleere Verweissammlungen entstehen. Durch Verweise auf inadäquate Dokumente kann ein falscher Eindruck entstehen.
- Hypertext ist unnatürlich (wie z.B. ein Film). Der Benutzer kann keinen Hypertext sprechen, sondern er braucht Hilfsmittel, um ihn vorzuführen.

Repräsentation

- Bestimmte Eigenschaften von traditionellen Textbeständen gehen verloren: Leder-Einband, Papierbeschaffenheit, Eselsohren, zwischen den Zeilen geschriebenes.

- Standards für die Darstellung/Repräsentation von Hypertext werden immer weiter entwikkelt.
- Inkonsistente Bestände (fehlerhafte Adressierung, Rechner/Netztausfälle, eingehende Beziehungen auf ein Dokument werden nicht verwaltet etc.) entstehen dadurch, daß konsistenzsichernde Maßnahmen für Hypertexte kaum verfügbar sind. Die Zuordnung verschiedener Typen auf Dokumente und Beziehungen wird bisher kaum unterstützt.
- Das Mehrautorenkonzept für Hypertext kann das Gesamtwerk verderben und eine Art Graffiti-Kultur erzeugen. Kohäsion, Kohärenz und Intention gehen verloren.
- Die vielfältigen Möglichkeiten von Hypertext lenken von dem zu sagenden, zeigenden ab.

Zugriff
- Rauschen: "*(...) einmal, daß schlicht zuviel über die Kanäle der neuen Medien reinströmt, und zum anderen, daß es Information gibt, die das Verstehen behindert oder verhindert.*" (Haefner nach Volpert (1985) S. 111)
- Hypertext kann nur über ein Werkzeug (z.B. Computer, Lesegerät etc.) benutzt werden. Diese Werkzeuge sind heute i.A. noch nicht so weit entwickelt, daß sie an jedem Ort eingesetzt werden können. Auch die Peripherie des Werkzeugs (z.B. Drucker, Scanner etc.) kann nicht an jedem Ort betrieben werden.
- Anfrageverfahren sind kaum vorhanden und nicht standardisiert, Standard-Benutzeroberflächen sind nicht in Sicht
- Die Übersichtlichkeit des Bestandes kann verloren gehen. Ein direkter "Griff" an die richtige Stelle ist u.U. schwierig.
- Der Autor schafft Irrgärten, der Benutzer betritt Irrgärten.
- Methoden zur Indexierung von Hypertextbeständen sind kaum standardisiert. Experten (Indexierer, Thesauruspfleger etc) werden bisher kaum im Bereich der Hypertexte eingesetzt.
- Ein einheitliches Veröffentlichungsschema existiert nicht.

Insgesamt kann die Frage, ob hypermediale Ausdrucksformen insgesamt ein Mehr oder ein Weniger von etwas bedeuten, nicht beantwortet werden. Manche Domänen eignen sich eher, manche weniger zur Darstellung/Benutzung als Hypertext. So wie es Bestsellerautoren sequentieller Texte gibt, wird es Bestsellerautoren von Hypertexten geben.

2.3 Graphentheorie

Es werden die Teilgebiete der Graphentheorie definiert, die in den Anwendungen (vgl. Kap. 4) benötigt werden (vgl. Schmidt, Ströhlein (1993)):
Definition 1: Ein *ungerichteter Graph* $G = (N, Adj)$ besteht aus einer Menge N von Knoten (nodes) und einer Relation $Adj \subseteq N \times N$, die irreflexiv und symmetrisch ist (Adjazenzrelation).
Definition 2: Ein *gerichteter Graph* $G = (N, Conn)$ besteht aus einer Menge N von Knoten (nodes) und einer Relation $Conn \subseteq N \times N$ (Assoziationsrelation).
Definition 3: Die *Transposition einer Relation* R wird wie folgt definiert:
$R^T := \{(x,y) \mid (y,x) \in R\}$
Definition 4: Wenn Conn eine Assoziationsrelation ist und x ein Knoten, dann ist Conn x die Menge der *direkten Vorgänger* von x .
Definition 5: Wenn Conn eine Assoziationsrelation ist und x ein Knoten, dann ist $Conn^T x$ die Menge der *direkten Nachfolger* von x .

Definition 6: Eine Sequenz von Knoten $s = (x_1, \ldots, x_n)$, $n > 0$ in einem gerichteten Graphen ist ein *Pfad von x_1 nach x_n* gdw. $x_{i-1} \subseteq \mathrm{Conn}\, x_i$ für $i = 1, \ldots, n$
Definition 7: Eine Sequenz von Knoten $s = (x_1, x_2, \ldots)$ von unendlicher Länge in einem gerichteten Graphen ist ein *Pfad unendlicher Länge beginnend bei x_1* gdw. $x_{i-1} \subseteq \mathrm{Conn}\, x_i$ für $i > 1$
Definition 8: Eine Sequenz von Knoten $s = (x_1, \ldots, x_n)$, $n > 0$ in einem gerichteten Graphen ist ein *Zyklus in x_1* gdw. S ist Pfad von x_1 nach x_n und $x_1 = x_n$
Definition 9: Ein Pfad oder Zyklus ist *elementar*, wenn kein Knoten der Sequenz mehr als einmal vorkommt.
Definition 10: Ein *zyklischer Graph* ist ein Graph der Zyklen enthält.
Definition 11: Wenn Conn eine Assoziationsrelation ist und x ein Knoten, dann ist $(\mathrm{Conn}^T (\mathrm{Conn}\, x))- \{x\}$ die Menge von *Geschwistern* von x .
Definition 12: $x_1, \ldots, x_n$ seien Knoten eines Graphen, Conn sei die Assoziationsrelation dieses Graphen. Dann führt die folgende Iteration zur *transitiven Hülle* $\mathrm{Conn}^+ = B_n$.
$B_0 := \mathrm{Conn}$, $B_i := B_{i-1} \cup B_{i-1}\, x_i\, x^T_i\, B_{i-1}$, für $i = 1, \ldots, n$
Der Aufwand bei diesem Verfahren liegt bei $O(n^3)$ booleschen Operationen (vgl. Roy, Warshall in Schmidt, Ströhlein (1993) S. 39).
Definition 13: Wenn Conn eine Assoziationsrelation ist und x ein Knoten, dann ist $\mathrm{Conn}^+\, x$ die Menge der *Vorgänger* (ancestors) von x.
Definition 14: Wenn Conn eine Assoziationsrelation ist und x ein Knoten, dann ist $\mathrm{Conn}^{+T}\, x$ die Menge der *Nachfolger* (descendants) von x.

2.4 Logik

Die mathematische Logik spielt in Informationssystemen wie Datenbanksystemen, Dokumentationssystemen, Hypertextsystemen, Frage-Antwort-Systemen und Expertensystemen eine wichtige Rolle. Die mathematische Formulierung von Problemen ist normalerweise kürzer, deutlicher und weniger anfällig für Fehlinterpretationen als die natürlichsprachliche Formulierung (vgl. Lyons (1980) S. 152).
In der mathematischen Logik wird die Aussagenlogik von der Prädikatenlogik unterschieden. Die Prädikatenlogik ist eine Erweiterung der Aussagenlogik. Wesentliche Erweiterungen der Prädikatenlogik sind Variablen, Funktoren, Prädikate und Quantoren. Die Individuen werden von den ihren Eigenschaften (Prädikaten) getrennt (vgl. Ackermann, Hilbert (1972) S. 67)
Beim Aufbau der Sprache[3] der Prädikatenlogik ist es wesentlich, die Syntax von der Semantik zu trennen.
"*Wenn nämlich erreicht werden kann, eine Sprache syntaktisch und semantisch getrennt aufzubauen, gilt unter bestimmten Voraussetzungen, daß sich alle Folgerungen aus einem Axiomensystem algorithmisch (syntaktisch) gewinnen lassen.*" (Reiner (1991) S. 10).
Anstatt mit einem einzigen Individuenbereich, können wir mit mehreren Gattungen von Individuenbereichen arbeiten. Wir gelangen so zur mehrsortigen Prädikatenlogik erster Stufe (vgl. Ackermann, Hilbert (1972) S. 117).
In dieser Arbeit ist die Prädikatenlogik Grundlage für den Aufbau der Anfragesprachen. Wir erweitern die Prädikatenlogik um Anfragen und verwenden den λ-Operator in λ-Ausdrücken, die Mengen von Gegenständen bezeichnen (Carnap (1958) S. 129ff, Konrad (1976)). Diese

[3] *"Sprechen heißt: in Tautologien verfallen. (...) - und auch ihre Widerlegung. Eine Zahl n möglicher Sprachen verwendet den gleichen Wortschatz; in einigen erlaubt das Symbol Bibliothek die korrekte Definition überall vorhandenes und fortdauerndes System sechseckiger Galerien, aber Bibliothek ist Brot oder Pyramide oder irgend etwas anderes, und die sieben Wörter, die sie definieren, haben einen anderen Bedeutungswert"*
(Eco 1987, 9)

Erweiterung gestattet uns Anwendungen wie in Kapitel 4, die Anfragemittel und Anfragegegenstand trennen.
Im Folgenden schreiben wir Sorten und Typen hochgestellt und Nummerierungen tiefgestellt hinter die jeweiligen Grundsymbole.

2.4.1 Prädikatenlogik 1. Stufe

Im Folgenden bauen wir die Sprache der Prädikatenlogik 1. Stufe auf. Wir erweitern die traditionelle Prädikatenlogik im Hinblick auf unsere Anwendungen (siehe Kap. 4). Mit Hilfe von λ-Ausdrücken können so Mengen von Gegenständen bezeichnet werden.

Syntax

Die Sprache der Prädikatenlogik 1. Stufe wird aus folgenden Grundsymbolen (Alphabet) aufgebaut:

Alphabet:
1. Individuenkonstanten: $c_1, c_2, ...$
Individuenvariablen: $x_1, x_2, ...$
2. Funktionskonstanten: $k_1, k_2, ...$
Funktionsvariablen: $f_1, f_2, ...$
3. Prädikatenkonstanten: $C_1, C_2, ...$
Prädikatenvariablen: $P_1, P_2, ...$
4. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation), $\leftrightarrow$ (Äquivalenz), $\exists$ (es gibt ein), $\forall$ (für alle), λ (die Menge)
5. Technische Symbole: (,)
Mit diesen Symbolen werden induktiv die Terme und Formeln gebildet:

Terme:
1. Individuenvariablen und -konstanten sind Terme.
2. Wenn $t_1, t_2, ... , t_n$ Terme sind und f ein Funktionssymbol (Konstante oder Variable) ist, dann ist $f(t_1, t_2, ... , t_n)$ ein Term.
3. Das sind alle Terme.

Formeln:
1. Wenn $t_1, t_2, ... , t_n$ Terme sind und P ein Prädikatensymbol (Konstante oder Variable) ist, dann ist $P(t_1, t_2, ... , t_n)$ eine Formel.
2. Wenn F, F_1 und F_2 Formeln sind, dann auch $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$, $(\exists x_i)F$, $(\forall x_i)F$.
3. Das sind alle Formeln.

Anfragen:
1. Wenn F eine Formel ist, dann ist (λx_i) F eine Anfrage.
2. Das sind alle Anfragen.

Semantik

In der Semantik wird den Ausdrücken einer Sprache eine Bedeutung gegeben. Dafür legen wir den nichtleeren Individuenbereich *U* (wie Universum; im Folgenden zur Unterscheidung grundsätzlich kursiv gesetzt) zugrunde.

Eine Interpretation $\Im$ der Sprache über U ist eine Abbildung, welche jedem Individuensymbol c_i bzw. x_i genau ein Element $\Im(c_i)$ bzw. $\Im(x_i)$ aus U, jedem Funktionssymbol k_i bzw. f_i genau eine Funktion $\Im(k_i)$ bzw. $\Im(f_i)$ von $U \times ... \times U$ auf U und jedem Prädikatensymbol C_i bzw. P_i genau eine Relation $\Im(C_i)$ bzw. $\Im(P_i)$ auf $U \times ... \times U$ zuordnet.
Jedem Term t kann nun ein Element $\Im(t)$ aus U zugeordnet werden. Für jede Formel kann erklärt werden, wann sie bei einer Interpretation über U gilt.

Interpretation der Individuen-, Funktions- und Prädikatensymbole
1. $\Im(c_1) \in U, \Im(c_2) \in U, ... , \Im(x_1) \in U, \Im(x_2) \in U, ... ,$
2. $\Im(k_i) \in \{U \times ... \times U \rightarrow U\}, \Im(f_i) \in \{U \times ... \times U \rightarrow U\},$
3. $\Im(C_i) \subseteq U \times ... \times U, \Im(P_i) \subseteq U \times ... \times U,$

Interpretation der Terme
4. $\Im(f(t_1, ... , t_n)) = \Im(f)(\Im(t_1), ... , \Im(t_n))$
wobei: f sei Funktionssymbol (Konstante oder Variable).

Interpretation der Formeln

5. $\Im$ ist Modell von $P(t_1, ... , t_n)$	gdw	$<\Im(t_1), ... , \Im(t_n)> \in \Im(P)$
wobei: P sei Prädikatensymbol (Konstante oder Variable).		
6. $\Im$ ist Modell von $\neg F$	gdw	$\Im$ ist nicht Modell von F.
$\Im$ ist Modell von $(F_1 \wedge F_2)$	gdw	$\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $(F_1 \vee F_2)$	gdw	$\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $(F_1 \rightarrow F_2)$	gdw	$\Im$ ist Modell von F2 wenn $\Im$ ist Modell von F_1.
$\Im$ ist Modell von $(F_1 \leftrightarrow F_2)$	gdw	($\Im$ ist Modell von F1 gdw $\Im$ ist Modell von F_2).
$\Im$ ist Modell von $(\exists x_i)$ F	gdw	$\Im^U_x$ ist Modell von F für mindestens ein $U \in U$.
$\Im$ ist Modell von $(\forall x_i)$ F	gdw	$\Im^U_x$ ist Modell von F für alle $U \in U$.

Anfragen
7. $\Im((\lambda x_i) F) = \{<U>: \Im^U_x$ ist Modell von F für $U \in U\}$.

2.4.2 Mehrsortige Prädikatenlogik 1. Stufe

In vielen Anwendungen haben wir es mit Individuen, Abbildungen und Relationen unterschiedlicher Sorten zu tun. Beispielsweise spricht man in der Geometrie von den Individuen „Punkt“ und „Gerade“ und der Abbildung „Projektion“, die einen Punkt auf eine Gerade abbildet und der Relation „Parallel“ die auf Geraden definiert ist.
Dafür wird die einfache Prädikatenlogik 1. Stufe um Sorten $s_1, s_2, ... , s_n$ erweitert. Individuen, Funktionen und Prädikate (Konstanten und Variablen) werden nach Sorten geordnet.

Syntax

Alphabet:
s sei eine bel. aber feste Sorte
1. Individuenkonstanten der Sorte s: $c^s_1, c^s_2, ...$
Individuenvariablen der Sorte s: $x^s_1, x^s_2, ...$
2. Funktionskonstanten von Sorten $s_1, ... , s_n$ auf die Sorte s: $k^{s1, ... , sn : s}_1, k^{s1, ... , sn : s}_2, ...$
Funktionsvariablen von Sorten $s_1, ... , s_n$ auf die Sorte s: $f^{s1, ... , sn : s}_1, f^{s1, ... , sn : s}_2, ...$
3. Prädikatenkonstanten der Sorten $s_1, ... , s_n$: $C^{s1, ... , sn}_1, C^{s1, ... , sn}_2, ...$

Prädikatenvariablen der Sorten $s_1, \ldots, s_n$: $P^{s1, \ldots, sn}{}_1$, $P^{s1, \ldots, sn}{}_2$, ...
4. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation), $\leftrightarrow$ (Äquivalenz), $\exists$ (es gibt ein), $\forall$ (für alle), λ (die Menge)
5. Technische Symbole: (,)
Bemerkung: Man kann die explizit hochgestellten Sorten eines Prädikats auch weglassen, da diese implizit durch die Sorten der Argumente gegeben sind. Weiterhin kann eine Funktion der Sorten $s_1, \ldots, s_n$ auf die Sorte s auch durch ein Prädikat der Sorten $s_1, \ldots, s_n$, s ausgedrückt werden.
Mit den Symbolen aus dem Alphabet werden induktiv die Terme und Formeln gebildet:

Terme:
1. Individuenkonstanten und -variablen der Sorte s sind Terme der Sorte s.
2. Wenn t_1 Term der Sorte s_1 , t_2 Term der Sorte s_2 ,... , t_n Term der Sorte s_n ist, und f ein Funktionssymbol (Konstante oder Variable) von Sorten $s_1, \ldots, s_n$ auf s ist, dann ist $f^{s1, \ldots, sn:s}(t_1, \ldots, t_n)$ Term der Sorte s.
3. Das sind alle Terme.

Formeln:
1. Wenn t_1 Term der Sorte s_1 , t_2 Term der Sorte s_2 ,... , t_n Term der Sorte s_n ist und P ein Prädikatensymbol (Konstante oder Variable) der Sorten $s_1, \ldots, s_n$ ist, dann ist $P^{s1, \ldots, sn}(t_1, t_2, \ldots, t_n)$ eine Formel.
2. Wenn F, F_1 und F_2 Formeln sind, dann sind auch $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$, $(\exists\, x^s{}_i)$ F, $(\forall\, x^s{}_i)$ F Formeln.
3. Das sind alle Formeln.

Anfragen:
1. Wenn F eine Formel ist, dann ist $(\lambda x^s{}_i)$ F eine Anfrage.
2. Das sind alle Anfragen.

Semantik

Eine Interpretation $\mathfrak{I}$ ist eine Abbildung, welche jedem Individuensymbol $c^s{}_i$ bzw. $x^s{}_i$ genau ein Element $\mathfrak{I}(c^s{}_i)$ bzw. $\mathfrak{I}(x^s{}_i)$ aus U^s, jedem Funktionssymbol $k^{s1, \ldots, sn:s}{}_i$ bzw. $f^{s1, \ldots, sn:s}{}_i$ genau eine Abbildung $\mathfrak{I}(k^{s1, \ldots, sn:s}{}_i)$ bzw. $\mathfrak{I}(f^{s1, \ldots, sn:s}{}_i)$ von $U^{s1} \times \ldots \times U^{sn}$ auf U^T und jedem Prädikatensymbol $C^{s1, \ldots, sn}{}_i$ bzw. $P^{s1, \ldots, sn}{}_i$ genau eine Relation $\mathfrak{I}(C^{s1, \ldots, sn}{}_i)$ bzw. $\mathfrak{I}(P^{s1, \ldots, sn}{}_i)$ auf $U^{s1} \times \ldots \times U^{sn}$ zuordnet.
Jedem Term t der Sorte s kann nun ein Element $\mathfrak{I}(t)$ aus U^s zugeordnet werden. Für jede Formel kann erklärt werden, wann sie bei einer Interpretation über $U^{s1} \times \ldots \times U^{sn}$ gilt.

Interpretation der Individuen-, Funktions- und Prädikatensymbole
1. $\mathfrak{I}(c^s{}_i) \in U^s$, $\mathfrak{I}(c^{s1}{}_i) \in U^{s1}$, ..., $\mathfrak{I}(x^s{}_i) \in U^s$, $\mathfrak{I}(x^{s1}{}_i) \in U^{s1}$, ... ,
2. $\mathfrak{I}(k^{s1, \ldots, sn:s}{}_i) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow_K U^s\}$, $\mathfrak{I}(f^{s1, \ldots, sn:s}{}_i) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow U^s\}$,
3. $\mathfrak{I}(C^{s1, \ldots, sn}{}_i) \subseteq U^{s1} \times \ldots \times U^{sn}$, $\mathfrak{I}(P^{s1, \ldots, sn}{}_i) \subseteq U^{s1} \times \ldots \times U^{sn}$,

Interpretation der Terme
4. $\mathfrak{I}(k^{s1, \ldots, sn:s}{}_i(t_1, \ldots, t_n)) = \mathfrak{I}(k^{s1, \ldots, sn:s}{}_i)\,(\mathfrak{I}(t_1), \mathfrak{I}(t_2), \ldots, \mathfrak{I}(t_n))$,
$\mathfrak{I}(f^{s1, \ldots, sn:s}{}_i(t_1, \ldots, t_n)) = \mathfrak{I}(f^{s1, \ldots, sn:s}{}_i)\,(\mathfrak{I}(t_1), \mathfrak{I}(t_2), \ldots, \mathfrak{I}(t_n))$,

Interpretation der Formeln

5. $\Im$ ist Modell von $C^{s1, \dots, sn}_i(t_1, \dots, t_n)$ gdw $<\Im(t_1), \dots, \Im(t_n)> \in \Im(C^{s1, \dots, sn}_i)$,
$\Im$ ist Modell von $P^{s1, \dots, sn}_i(t_1, \dots, t_n)$ gdw $<\Im(t_1), \dots, \Im(t_n)> \in \Im(P^{s1, \dots, sn}_i)$,
6. $\Im$ ist Modell von $\neg F$ gdw $\Im$ ist nicht Modell von F.
$\Im$ ist Modell von $(F_1 \wedge F_2)$ gdw $\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $(F_1 \vee F_2)$ gdw $\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $(F_1 \rightarrow F_2)$ gdw $\Im$ ist Modell von F2 wenn $\Im$ ist Modell von F_1.
$\Im$ ist Modell von $(F_1 \leftrightarrow F_2)$ gdw ($\Im$ ist Modell von F1 gdw $\Im$ ist Modell von F_2).
$\Im$ ist Modell von $(\exists\, x^s_i)$ F gdw $\Im^U_{xi}$ ist Modell von F für mindestens ein $U \in \mathcal{U}^s$.
$\Im$ ist Modell von $(\forall\, x^s_i)$ F gdw $\Im^U_{xi}$ ist Modell von F für alle $U \in \mathcal{U}^s$.

Interpretation der Anfragen
7. $\Im((\lambda\, x^s_i)\, F) = \{<U>: \Im^U_{xi}$ ist Modell von F für $U \in \mathcal{U}^s\}$.

2.4.3 Mehrstufige Prädikatenlogik

Die Prädikatenlogik 1. Stufe wird zur mehrstufigen Prädikatenlogik (vgl. Carnap (1958) S. 80ff und Ackermann, Hilbert (1972) S. 164) erweitert.
Um sogenannte logische Antinomien zu vermeiden, führte Bertand Russell Typen ein.

Syntax

Der Typ wird induktiv definiert:
1. 0 ist ein Typ.
2. Sind $T_1, \dots, T_n$ Typen, dann ist auch $(T_1, \dots, T_n)$ ein Typ.
Stufe: Die Stufe eines Ausdrucks ergibt sich aus der maximalen Anzahl von umgebenden Klammerpaaren einer der 0 des Typs dieses Ausdrucks. Beispielsweise ist ein Ausdruck des Typs ((0,0),0) ein Ausdruck der 2. Stufe.
Der Typ wird im Folgenden hochgestellt hinter die Symbole und Ausdrücke geschrieben.

Alphabet
1. Individuenkonstanten: $c_1, c_2, \dots$
Individuenvariablen: $x_1, x_2, \dots$
2. Funktionskonstanten von Typen $T_1, \dots, T_n$ auf den Typ T: $k^{T1, \dots, Tn : T}_1, k^{T1, \dots, Tn : T}_2, \dots$
Funktionsvariablen von Typen $T_1, \dots, T_n$ auf den Typ T: $f^{T1, \dots, Tn : T}_1, f^{T1, \dots, Tn : T}_2, \dots$
3. Prädikatenkonstanten der Typen $T_1, \dots, T_n$: $C^{T1, \dots, Tn}_1, C^{T1, \dots, Tn}_2, \dots$
Prädikatenvariablen der Typen $T_1, \dots, T_n$: $P^{T1, \dots, Tn}_1, P^{T1, \dots, Tn}_2, \dots$
4. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation), $\leftrightarrow$ (Äquivalenz),
$\exists$ (es gibt ein), $\forall$ (für alle), λ (die Menge)
5. Technische Symbole: (,)

Ausdrücke
1. Individuenkonstanten und –variablen sind Ausdrücke vom Typ 0. Funktionskonstanten und –variablen sind Ausdrücke vom Typ (0, ... (n-mal) ... , 0 : 0). Prädikatenkonstanten und –variablen sind Ausdrücke vom Typ (0, ... (n-mal) ... , 0).
2. Wenn A_1 ein Ausdrucks des Typs T_1 , A_2 Ausdruck des Typs T_2 ,... , A_n Ausdruck des Typs T_n ist und $f^{(T1, \dots, Tn:T)}$ ein Funktionssymbol (Konstante oder Variable) vom Typ T_1 ,... , T_n auf den Typ T ist, dann ist $f^{(T1, \dots, Tn:T)}(A_1, \dots, A_n)$ ein Funktionsausdruck des Typs T.
3. Wenn A_1 ein Ausdrucks des Typs T_1 , A_2 Ausdruck des Typs T_2 ,... , A_n Ausdruck des Typs T_n ist und $P^{(T1, \dots, Tn)}$ ein Prädikatensymbol (Konstante oder Variable) des Typs T_1 ,... , T_n ist, dann ist $P^{(T1, \dots, Tn)}(A_1, \dots, A_n)$ ein Prädikatausdruck des Typs T_1 ,... , T_n.

4. Wenn A, A_1 und A_2 Ausdrücke sind, dann sind auch $\neg A$, $A_1 \wedge A_2$, $A_1 \vee A_2$, $A_1 \rightarrow A_2$, $A_1 \leftrightarrow A_2$, $(\exists\, x_i)$ A, $(\forall\, x_i)$ A, $(\exists\, P^{(T1, \dots, Tn)}{}_i)$ A, $(\forall\, P^{(T1, \dots, Tn)}{}_i)$ A Formeln.
5. Das sind alle Formeln.

Bemerkung: $P_1(f_1(c_1, c_2))$, $P_1(P_2, P_3)$, oder $f_1(P_1, P_2, f_2)$ sind beispielsweise Ausdrücke, aber $P_1(P_2(c_1, c_2))$ nicht.

Anfragen
1. Wenn A ein Ausdruck ist, dann ist (λx_i) A eine Anfrage.
2. Wenn A ein Ausdruck ist, dann ist $(\lambda P^{(T1, \dots, Tn)}{}_i)$ A eine Anfrage.

Semantik

Eine Interpretation $\Im$ ist eine Abbildung, welche jedem Individuensymbol c_i bzw. x_i genau ein Element $\Im(c_i)$ bzw. $\Im(x_i)$ aus U, jedem Funktionssymbol $k^{T1, \dots, Tn:T}{}_i$ bzw. $f^{T1, \dots, Tn:T}{}_i$ genau eine Abbildung $\Im(k^{T1, \dots, Tn:T}{}_i)$ bzw. $\Im(f^{T1, \dots, Tn:T}{}_i)$ von $U^{T1} \times \dots \times U^{Tn}$ auf U^T und jedem Prädikatensymbol $C^{T1, \dots, Tn}{}_i$ bzw. $P^{T1, \dots, Tn}{}_i$ genau eine Relation $\Im(C^{T1, \dots, Tn}{}_i)$ bzw. $\Im(P^{T1, \dots, Tn}{}_i)$ auf $U^{T1} \times \dots \times U^{Tn}$ zuordnet.

Interpretation der Individuen- Funktions- und Prädikatensymbole
1. $\Im(c_i) \in U$, $\Im(x_i) \in U$,
2. $\Im(k^{T1, \dots, Tn:T}{}_i) \in \{U^{T1} \times \dots \times U^{Tn} \rightarrow_K U^T\}$, $\Im(f^{T1, \dots, Tn:T}{}_i) \in \{U^{T1} \times \dots \times U^{Tn} \rightarrow U^T\}$,
3. $\Im(C^{(T1, \dots, Tn)}{}_i) \subseteq U^{T1} \times \dots \times U^{Tn}$, $\Im(P^{(T1, \dots, Tn)}{}_i) \subseteq U^{T1} \times \dots \times U^{Tn}$,

Interpretation der Funktionsausdrücke
4. $\Im(k^{(T1, \dots, Tn:T)}{}_i(A_1, \dots, A_n)) = \Im(k^{(T1, \dots, Tn:Tm)}{}_i)\, (\Im(A_1), \dots, \Im(A_n))$,
$\Im(f^{(T1, \dots, Tn:T)}{}_i(A_1, \dots, A_n)) = \Im(f^{(T1, \dots, Tn:T)}{}_i)\, (\Im(A_1), \dots, \Im(A_n))$,

Interpretation der Prädikatausdrücke
5. $\Im$ ist Modell von $C^{(T1, \dots, Tn)}{}_i(A_1, \dots, A_n)$ gdw $\langle\Im(A_1), \dots, \Im(A_n)\rangle \in \Im(C^{(T1, \dots, Tn)}{}_i)$,
$\Im$ ist Modell von $P^{(T1, \dots, Tn)}{}_i(A_1, \dots, A_n)$ gdw $\langle\Im(A_1), \dots, \Im(A_n)\rangle \in \Im(P^{(T1, \dots, Tn)}{}_i)$,
6. $\Im$ ist Modell von $\neg A$ gdw $\Im$ ist nicht Modell von F.
$\Im$ ist Modell von $(A_1 \wedge A_2)$ gdw $\Im$ ist Modell von A1 und $\Im$ ist Modell von A_2.
$\Im$ ist Modell von $(A_1 \vee A_2)$ gdw $\Im$ ist Modell von A1 oder $\Im$ ist Modell von A_2.
$\Im$ ist Modell von $(A_1 \rightarrow A_2)$ gdw $\Im$ ist Modell von A2 wenn $\Im$ ist Modell von A_1.
$\Im$ ist Modell von $(A_1 \leftrightarrow A_2)$ gdw ($\Im$ ist Modell von A1 gdw $\Im$ ist Modell von A_2).
$\Im$ ist Modell von $(\exists\, x_i)$ A gdw $\Im^U{}_{xi}$ ist Modell von A für mindestens ein $U \in U$.
$\Im$ ist Modell von $(\forall\, x_i)$ A gdw $\Im^U{}_{xi}$ ist Modell von A für alle $U \in U$.
$\Im$ ist Modell von $(\exists\, P^{(T1, \dots, Tn)}{}_i)$ A gdw
$\Im^{rel}{}_{Pi}$ ist Modell von A für mindestens eine Relation über $U^{T1} \times \dots \times U^{Tn}$.
$\Im$ ist Modell von $(\forall\, P^{(T1, \dots, Tn)}{}_i)$ A gdw
$\Im^{rel}{}_{Pi}$ ist Modell von A für alle Relationen über $U^{T1} \times \dots \times U^{Tn}$.

Interpretation der Anfragen
7. $\Im((\lambda\, x_i)\, A) = \{\langle U\rangle: \Im^U{}_{xi}$ ist Modell von A für $U \in U\}$.
$\Im((\lambda\, C^{(T1, \dots, Tn)}{}_i)\, A) = \{\langle rel\rangle: \Im^{rel}{}_{Ci}$ ist Modell von A$\}$.
$\Im((\lambda\, P^{(T1, \dots, Tn)}{}_i)\, A) = \{\langle rel\rangle: \Im^{rel}{}_{Pi}$ ist Modell von A$\}$.

Beispiel

Individuenbereich:
$U = \text{NAT} = \{0, 1, 2, 3, ...\}$

Relationen:
$\text{LT} = \{<0,1>,<0,2>,<0,3>, ... , <1,2>, <1,3>,<1,4>, ... ,<2,3>, <2,4>, ...\}$
$\text{TRANS} =\{<\text{LT}>\}=\{<\{<0, 1>, <0, 2>, <0, 3>, ... ,<1, 2>,<1, 3>,<1, 4>,...,<2, 3>,<2,4>,...\}>\}$

In der Sprache der mehrstufigen Prädikatenlogik wird definiert:
'1', '2', '3', ... seien Individuenkonstanten, $\text{Lessthan}^{(0,0)}$ sei eine Prädikatenkonstante vom Typ (0,0), $P^{(0,0)}$ sei eine Prädikatenvariable vom Typ (0,0), $P^{((0,0))}$ sei eine Prädikatenvariable vom Typ ((0,0)),

1. Welche Beziehungsarten bestehen zwischen den beiden natürlichen Zahlen '1' und '2' ?

$((\lambda\, P^{(0,\,0)})\, P^{(0,\,0)}\, (\text{'1'}, \text{'2'})\,)$
$= \{<\text{rel}> : \mathfrak{I}^{\text{rel}}{}_{P} \text{ ist Modell von } P^{(0,\,0)}\, (\text{'1'}, \text{'2'})\}$
$= \{<\text{rel}> : <\mathfrak{I}^{\text{rel}}{}_{P}\, (\text{'1'}), \mathfrak{I}^{\text{rel}}{}_{P}\, (\text{'2'})> \in \mathfrak{I}^{\text{rel}}{}_{P}(P^{(0,\,0)})\}$
$= \{<\text{rel}> : <1, 2> \in \mathfrak{I}^{\text{rel}}{}_{P}(P^{(0,\,0)})\}$
$= \{<\text{rel}> : <1, 2> \in (\text{NAT} \times \text{NAT})$
$= \{<\text{LT}>\}$

2. Welche Eigenschaften hat die Beziehungsart Lessthan ?

$((\lambda\, P^{((0,\,0))})\, P^{((0,\,0))}\, (\text{Lessthan}^{(0,\,0)}\,)\,)$
$= \{<\text{rel}> : \mathfrak{I}^{\text{rel}}{}_{P} \text{ ist Modell von } P^{((0,\,0))}\, (\text{Lessthan}^{(0,\,0)})\}$
$= \{<\text{rel}> : <\mathfrak{I}^{\text{rel}}{}_{P}\, (\text{Lessthan}^{(0,\,0)})> \in \mathfrak{I}^{\text{rel}}{}_{P}(P^{((0,\,0))})\}$
$= \{<\text{rel}> : <\{<0, 1>, <1, 2>, <2, 3>, ...\}> \in ((\text{NAT} \times \text{NAT}))$
$= \{<\text{TRANS}>\}$

2.4.4 Mehrstufige mehrsortige Prädikatenlogik

Syntax

Wir definieren induktiv den Typ:
1. Die Sorten $s_1, ... , s_n$ sind Typen.
2. Sind $T_1, ... , T_n$ Typen, dann ist auch $(T_1, ... , T_n)$ ein Typ.

Stufe: Die Stufe eines Ausdrucks ergibt sich aus der maximalen Anzahl von umgebenden Klammerpaaren einer der s_i des Typs dieses Ausdrucks. Beispielsweise ist ein Ausdruck des Typs $((s_1, s_2), s_2)$ ein Ausdruck der 2. Stufe.

Alphabet

s sei eine bel. aber feste Sorte
1. Individuenkonstanten der Sorte s: $c^s{}_1, c^s{}_2, ...$
Individuenvariablen der Sorte s: $x^s{}_1, x^s{}_2, ...$
2. Funktionskonstanten von Typen $T_1, ... , T_n$ auf den Typ T: $k^{T1,\, ...\, ,\, Tn:T}{}_1, k^{T1,\, ...\, ,\, Tn:T}{}_2, ...$
Funktionsvariablen von Typen $T_1, ... , T_n$ auf den Typ T: $f^{T1,\, ...\, ,\, Tn:T}{}_1, f^{T1,\, ...\, ,\, Tn:T}{}_2, ...$
3. Prädikatenkonstanten der Typen $T_1, ... , T_n$: $C^{T1,\, ...\, ,\, Tn}{}_1, C^{T1,\, ...\, ,\, Tn}{}_2, ...$
Prädikatenvariablen der Typen $T_1, ... , T_n$: $P^{T1,\, ...\, ,\, Tn}{}_1, P^{T1,\, ...\, ,\, Tn}{}_2, ...$
4. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation), $\leftrightarrow$ (Äquivalenz), $\exists$ (es gibt ein), $\forall$ (für alle), λ (die Menge)

5. Technische Symbole: (,)

Ausdrücke
1. Individuenkonstanten und –variablen der Sorte s sind Ausdrücke vom Typ s. Funktionskonstanten und –variablen von Typen $T_1, ... , T_n$ auf T sind Ausdrücke vom Typ $T_1, ..., T_n{:}T$. Prädikatenkonstanten und –variablen der Typen $T_1, ... , T_n$ sind Ausdrücke vom Typ $T_1, ... , T_n$.
2. Wenn A_1 ein Ausdrucks des Typs T_1 , A_2 Ausdruck des Typs T_2 ,... , A_n Ausdruck des Typs T_n ist und $f^{(T1, ... , Tn:T)}$ ein Funktionssymbol (Konstante oder Variable) vom Typ T_1 ,... , T_n auf den Typ T ist, dann ist $f^{(T1, ... , Tn:T)}(A_1 ,... , A_n)$ ein Funktionsausdruck des Typs T.
3. Wenn A_1 ein Ausdrucks des Typs T_1 , A_2 Ausdruck des Typs T_2 ,... , A_n Ausdruck des Typs T_n ist und $P^{(T1, ... , Tn)}$ ein Prädikatensymbol (Konstante oder Variable) des Typs T_1 ,... , T_n ist, dann ist $P^{(T1, ... , Tn)}(A_1 ,... , A_n)$ ein Prädikatausdruck des Typs T_1 ,... , T_n.
4. Wenn A, A_1 und A_2 Ausdrücke sind, dann sind auch $\neg A$, $A_1 \wedge A_2$, $A_1 \vee A_2$, $A_1 \rightarrow A_2$, $A_1 \leftrightarrow A_2$, $(\exists x_i)$ A, $(\forall x_i)$ A $(\exists P^{(T1, ... , Tn)}{}_i)$ A, $(\forall P^{(T1, ... , Tn)}{}_i)$ A Formeln.
5. Das sind alle Formeln.

Anfragen
1. Wenn A ein Ausdruck ist, dann ist $(\lambda x^s{}_i)$ A eine Anfrage.
2. Wenn A ein Ausdruck ist, dann ist $(\lambda C^{(T1, ... , Tn)}{}_i)$ A eine Anfrage.
3. Wenn A ein Ausdruck ist, dann ist $(\lambda P^{(T1, ... , Tn)}{}_i)$ A eine Anfrage.

Semantik

Eine Interpretation $\Im$ ist eine Abbildung, welche jedem Individuensymbol c_i bzw. x_i genau ein Element $\Im(c_i)$ bzw. $\Im(x_i)$ aus U, jedem Funktionssymbol $k^{T1, ... , Tn:T}{}_i$ bzw. $f^{T1, ... , Tn:T}{}_i$ genau eine Abbildung $\Im(k^{T1, ... , Tn:T}{}_i)$ bzw. $\Im(f^{T1, ... , Tn:T}{}_i)$ von $U^{T1} \times ... \times U^{Tn}$ auf U^T und jedem Prädikatensymbol $C^{T1, ... , Tn}{}_i$ bzw. $P^{T1, ... , Tn}{}_i$ genau eine Relation $\Im(C^{T1, ... , Tn}{}_i)$ bzw. $\Im(P^{T1, ... , Tn}{}_i)$ auf $U^{T1} \times ... \times U^{Tn}$ zuordnet.

Interpretation der Individuen- Funktions- und Prädikatensymbole
1. $\Im(c^s{}_i) \in U^s$, $\Im(c^{s1}{}_i) \in U^{s1}$, ..., $\Im(x^s{}_i) \in U^s$, $\Im(x^{s1}{}_i) \in U^{s1}$, ... ,
2. $\Im(k^{T1, ... , Tn:T}{}_i) \in \{U^{T1} \times ... \times U^{Tn} \rightarrow_K U^T\}$, $\Im(f^{T1, ... , Tn:T}{}_i) \in \{U^{T1} \times ... \times U^{Tn} \rightarrow U^T\}$,
3. $\Im(C^{(T1, ... , Tn)}{}_i) \subseteq U^{T1} \times ... \times U^{Tn}$, $\Im(P^{(T1, ... , Tn)}{}_i) \subseteq U^{T1} \times ... \times U^{Tn}$,

Interpretation der Funktionsausdrücke
4. $\Im(k^{(T1, ... , Tn:T)}{}_i(A_1, ... , A_n)) = \Im(k^{(T1, ... , Tn:Tm)}{}_i)\ (\Im(A_1), ... , \Im(A_n))$,
$\Im(f^{(T1, ... , Tn:T)}{}_i(A_1, ... , A_n)) = \Im(f^{(T1, ... , Tn:T)}{}_i)\ (\Im(A_1), ... , \Im(A_n))$,

Interpretation der Prädikatausdrücke
5. $\Im$ ist Modell von $C^{(T1, ... , Tn)}{}_i(A_1, ... , A_n)$ gdw $<\Im(A_1), ... , \Im(A_n)> \in \Im(C^{(T1, ... , Tn)}{}_i)$,
$\Im$ ist Modell von $P^{(T1, ... , Tn)}{}_i(A_1, ... , A_n)$ gdw $<\Im(A_1), ... , \Im(A_n)> \in \Im(P^{(T1, ... , Tn)}{}_i)$,
6. $\Im$ ist Modell von $\neg A$ gdw $\Im$ ist nicht Modell von F.
$\Im$ ist Modell von $(A_1 \wedge A_2)$ gdw $\Im$ ist Modell von A1 und $\Im$ ist Modell von A_2.
$\Im$ ist Modell von $(A_1 \vee A_2)$ gdw $\Im$ ist Modell von A1 oder $\Im$ ist Modell von A_2.
$\Im$ ist Modell von $(A_1 \rightarrow A_2)$ gdw $\Im$ ist Modell von A2 wenn $\Im$ ist Modell von A_1.
$\Im$ ist Modell von $(A_1 \leftrightarrow A_2)$ gdw ($\Im$ ist Modell von A1 gdw $\Im$ ist Modell von A_2).
$\Im$ ist Modell von $(\exists x_i)$ A gdw $\Im^U{}_{xi}$ ist Modell von A für mindestens ein $U \in U$.
$\Im$ ist Modell von $(\forall x_i)$ A gdw $\Im^U{}_{xi}$ ist Modell von A für alle $U \in U$.
$\Im$ ist Modell von $(\exists P^{(T1, ... , Tn)}{}_i)$ A gdw

$\mathfrak{I}^{rel}_{Pi}$ ist Modell von A für mindestens eine Relation über $U^{T1} \times ... \times U^{Tn}$.
$\mathfrak{I}$ ist Modell von $(\forall P^{(T1, ..., Tn)}_{i})$ A gdw
$\mathfrak{I}^{rel}_{Pi}$ ist Modell von A für alle Relationen über $U^{T1} \times ... \times U^{Tn}$.

Interpretation der Anfragen
7. $\mathfrak{I}((\lambda x^{s}_{i}) A) = \{\langle U \rangle : \mathfrak{I}^{U}_{xi}$ ist Modell von A für $U \in U^{s}\}$.
$\mathfrak{I}((\lambda C^{(T1, ..., Tn)}_{i}) A) = \{\langle rel \rangle : \mathfrak{I}^{rel}_{Ci}$ ist Modell von A$\}$.
$\mathfrak{I}((\lambda P^{(T1, ..., Tn)}_{i}) A) = \{\langle rel \rangle : \mathfrak{I}^{rel}_{Pi}$ ist Modell von A$\}$.

Bemerkung: Man beachte, daß z.B. gilt:
$\langle U^{s1}_{1}, U^{s2}_{1} \rangle \in (U^{s1} \times U^{s2})$, $\langle\langle U^{s1}_{1}, U^{s2}_{1} \rangle, U^{s3}_{1} \rangle \in ((U^{s1} \times U^{s2}) \times U^{s3})$ oder
$\langle \{ \langle U^{s1}_{1}, U^{s2}_{3} \rangle, \langle U^{s1}_{2}, U^{s2}_{1} \rangle, \langle U^{s1}_{3}, U^{s2}_{2} \rangle \} \rangle \in ((U^{s1} \times U^{s2}))$.

2.5 Anfragesprachen

Informationssuche ist die Suche nach Information mittels einer Sprache. Diese Sprache wird als Anfragesprache[4] bezeichnet. Anfragesprachen werden unterschieden nach dem Zweck ihres Einsatzes oder dem Schwierigkeitsgrad ihrer Benutzung. Es werden Anfragesprachen für ungeübte und geübte Benutzer, für Retrievalexperten, systemnahe Anfragesprachen und Systemsprachen unterschieden. Anfragesprachen können ineinander übersetzt werden. So kann z.B. eine Anfragesprache für ungeübte Benutzer in eine systemnahe Zwischensprache und diese wiederum in die Systemsprache übersetzt werden. Ein hierarchischer Aufbau der Anfragesprachen ergibt sich, wenn die Anfragesprachen nach dem Expertengrad der Benutzer ineinander übersetzt werden können, z.B. von den Sprachen für ungeübte Benutzer hin zu den Systemsprachen. Reiner (1991) baut beispielsweise mit IQL (Intermediary Query Language) eine Anfragesprache zur Suche nach Dokumenten, Fakten und Erklärungen auf. Diese Sprache ist als Zwischensprache zwischen Benutzeranfragesprache und Systemsprache konzipiert. Reiner spezifiziert darüberhinaus Sprachen für Retrievalexperten und ungeübte Nutzer.
Besonders interessant sind Anfragesprachen oberhalb der Ebene der Systemsprachen. Die Informationssuche mit einer solchen Anfragesprache ist unabhängig vom verwendeten Informationssystem[5].

Erster Schritt bei der Entwicklung einer Anfragesprache ist die Analyse der zugrundeliegenden Dokumentenbestände (vgl. Kap. 3). In einem zweiten Schritt wird die Anfra

---

[4] *"Wir nennen diese Sprache Anfragesprache. Mit diesem Terminus soll zum Ausdruck kommen, daß der Anfragende (...) in dieser Sprache Fragen an das Informationssystem stellen kann. Der Terminus läßt offen, wonach gefragt wird. (...) Weitverbreitet ist der Terminus Abfragesprache, der den Abrufprozeß des Gedächtnisses (Speichers) betont.“* (Reiner 1991, 7/8)

[5] *"Von besonderem Interesse sind in diesem Zusammenhang allgemeine Systemschnittstellen, die es dem Nutzer erlauben, seine Suchanfrage unabhängig vom System in einer einheitlichen Suchsprache zu formulieren. Die Systemschnittstelle sollte dann diese allgemeine Suchsprache in die unterschiedlichen Systemsprachen der einzelnen Informationssysteme übersetzen (...). Systemübergreifende Benutzerschnittstellen sind deshalb äußerst nützlich, weil sie dem Nutzer die Mühe abnehmen, sich mit internen Details der einzelnen Informationssysteme auseinanderzusetzen. Von weit größerem Interesse sind Systeme, die selbst entscheiden, welche Informationsquelle bei einer bestimmten Anfrage abzufragen ist. (...) Der Entwurf eines solchen intelligenten Informationssystems ist zur Zeit noch nicht möglich."*
(Salton, McGill 1987, 454)

gesprache mit Hilfe einer Methode aufgebaut, die eine Trennung von Syntax und Semantik[6] vorsieht (vgl. Kutschera (1975) S. 223 und Konrad, Reiner (1985), Reiner (1991)): Welches Vokabular steht für die Anfrage zur Verfügung und in welcher Kombination kann es benutzt werden (Syntax)? Welche Interpretationsvorschriften werden gegeben? Was bedeutet eine syntaktisch korrekt gebildete Anfrage bzw. was liefert sie als Ergebnis zurück? Wie ist also das Verhältnis der Anfragezeichen zur "Wirklichkeit" definiert (Semantik)? Dabei können Zeichensätze umcodiert werden (beispielsweise das 8-Bit-Zeichen "ü" in die beiden 7-Bit-Zeichen "ue" oder umgekehrt). Eine Anfragesprachen exakt zu definieren heißt ihre Syntax und Semantik exakt zu spezifizieren.

Anfragesprachen können nach dem Dokumenttyp in folgende Klassen eingeteilt werden:
1) Anfragesprachen zur Suche in einfach strukturierten Dokumenten (incl. Volltextsuche)
2) Anfragesprachen zur Suche in link-strukturierten Dokumenten
3) Anfragesprachen zur Suche in geschachtelten Dokumenten
4) Anfragesprachen zur Faktensuche
5) Anfragesprachen zur Suche nach terminologischen Einträgen
Beispiele für (1) sind IQL (Reiner (1991)), Messenger (Messenger (1995), STN (1997)), Z39.50 (Z39.50-1995), WAIS (FreeWAIS (1995), CNIDR (1996)), Harvest (Camargo(1994)), Aliweb (2000), AltaVista (2000), Excite (2000), Fireball (2000), Google (2000), Harvest (2000), InfoSeek (2000), Inktomi (2000), Kolibri (2000), Lycos (2000), MetaCrawler (2000), OpenText (1997), Webcrawler (2000), Yahoo (2000).
Beispiele für (2) sind QBE (Zloof (1976)), G und G+ (Cruz, Mendelzon, Wood (1987), Wood (1988)), Reimer (1991)), HyQ (DeRose, Durand (1994), ISO 10744), HTTP (RFC 1068), Hyper-G (Kappe, Pani (1996)), UnQL (Fernandez , Popa , Suciu (1997)), W3QS (Konopnicki, Shmueli (1998), W3QS (1999)), WebSQL (WebSQL (1999)), Clever (Clever (2001)).
Beispiele für (3) sind OQL (Cattell (1994)), SQL 3 (ISO-SQL3), Lorel/TSIMMIS (Abiteboul et.al. (1997), TSIMMIS (1999)), XQL (Lapp, Robie, Schach (1998)), XSL, Xpath, Xpointer, Xlink (alle siehe Goldfarb, Prescod (2001)).
Beispiele für (4) sind SQL (ISO 9075, DIN 66315, ISO-SQL3), OQL (Cattell (1994)), SQL 3 (ISO-SQL3).
Beispiele für (5) sind CCL (ISO 8777), IQL-DIT (Reiner (1991) S. 107), Messenger (Messenger (1995), STN (1997)), THW^{3}Query (Hartlep (1996)), Cyc (Guha, Lenat (1990a), Guha, Lenat (1990b), Cyc (1997)).

Weiterhin sind logische Anfragesprachen zu unterscheiden, die z.T. einen übergreifenden Charakter darstellen: KL-ONE (Brachmann, Schmolze (1985), Baader et.al.(1990)), DATALOG (Ceri, Gottlob, Tanca (1990), Fuhr (1995)), Afrati-Koutras-Modell (Afrati, Koutras (1990)), Beeri-Kornatzky-Modell (Beeri, Kornatzky (1990)), Cyc (Guha, Lenat (1990a)), IQL (Reiner (1991)), WebLog (Lakshmanan, Sadri, Subramanian (1996)).

Man kann allgemein sagen, daß eine Anfragesprache A mächtiger als eine Anfragesprache B ist, wenn A den Dokumentenbestand schärfer trennt als B.
Eine Anfragesprache A ist mächtiger als eine Anfragesprache B, wenn die Menge der in A bildbaren Suchergebnismengen, die Menge der in B bildbaren Suchergebnismengen umfaßt. (Adaption aus Konrad (1986) S. 566).

[6] "*Die Syntax dieser Kunstsprachen (...) ist nach der Idee der charakteristica universalis von Leibniz in so enger Entsprechung zur Semantik aufgebaut, daß die syntaktische Form der Ausdrücke die Struktur ihrer Bedeutungen widerspiegelt.*" (Kutschera 1975, 222)

Die mächtigste denkbare Anfragesprache könnte die Potenzmenge des Dokumentenbestands (Menge aller Teilmengen des Dokumentenbestands) bilden. Diese Sprache ist jedoch im praktischen Einsatz ungeeignet, da sie für die unterschiedlichen Benutzeranfragesprachen zu mächtige Sprachmittel bereitstellt. Systemsprachen müßten diese unnötige Mächtigkeit mit dem Preis einer geringeren Systemperformanz im Informationssystem nachbilden.

3 Anfragen in Internet-Informationssystemen

In Internet-Informationssystemen stehen Darstellungs- und Navigationsaspekte im Vordergrund. Anfragen werden explizit nur am Rande behandelt. Man kann von Glück reden, wenn Teile der Syntax der Anfragen oder sogar die Semantik definiert werden. Positive Beispiele sind die Anfragesprache HyQ (ISO 10744, DeRose, Durand (1994)) und XQL (Lapp, Robie, Schach (1998)), bei der die Syntax definiert und Anfragebeispiele gegeben werden.
Besser entwickelt sind Anfrageverfahren, die sich hinter den Benutzeroberflächen (inzwischen meist Fenstersysteme) der einzelnen Systeme verbergen. Ein Mausklick kann zum Beispiel eine Nachbareinheit eines Dokuments oder ein drei Schritte zuvor gefundendes Dokument selektieren oder aktivieren. Eine Bewegung des Scrollbuttons am rechten Rand des Fensters führt dazu, daß weitere Dokumente geladen werden. Oder mit dem Drag&Drop-Mechanismus werden Teilbereiche von einem Teil des Fensters in einen anderen Teil bewegt.

Im Folgenden wird eine Untersuchung der Anfragemöglichkeiten häufig verwendeter Internet-Informationssysteme durchgeführt. Es werden Systemklassen nach den Dokumenttypen gebildet (Suche in einfach strukturierten Dokumenten, Suche in link-strukturierten Dokumenten, Suche in geschachtelten Dokumenten, Faktensuche, Suche nach terminologischen Einträgen und intelligente Agenten). Für jede Systemklasse, teilweise auch für die einzelnen Systeme, werden Anfragemöglichkeiten beschrieben und Anfragebeispiele gegeben.
Im letzten Teilkapitel wird die Untersuchung zusammengefaßt. Dabei werden die Kriterien Erfassung, Bestand, Suchmittel, Suchergebnisse und Leistungsgrenzen unterschieden.
Die Untersuchung kann nicht den letzten Stand der Entwicklung berücksichtigen, da eine große Entwicklungsgeschwindigkeit zu beobachten ist. Auch können nicht alle Dokumenttypen (z.B. 3-D-Objekte etc.) behandelt werden.
Teile der Untersuchung fließen in den Aufbau von Anfragesprachen für Internet-Informationssysteme ein (siehe Kap. 4).

3.1 Suche in einfach strukturierten Dokumenten

3.1.1 Personen- und Gruppensuche

Erste Personeninformationssysteme im Internet sind die Unix-Programme Rusers (Handschuch (1997)), Rwho (Handschuch (1995)) und Finger (RFC 1288).
Rusers, Rwho und Finger ermöglichen die Suche nach Personen, die eine Benutzerkennung in einem Rechnernetz besitzen. Folgende Felder für Personen werden bei Finger unterschieden:

- Benutzerkennung
- Vor- und Nachname
- Zeitpunkt der letzten Tätigkeit am Rechner
- Zeitpunkt des letzten Logins
- Information über bearbeitete Projekte
- Persönliche Daten (Telefonnummer, Hobbies etc.)

Es können die momentan eingelogten Benutzer erfragt werden. Nachteil ist, daß der Bestand auf jeweils einen Rechner beschränkt ist. Der Anwender muß deshalb wissen, auf welchem Rechner er die gesuchte Person vermutet. Weiterhin ist ein Zugriff häufig nicht möglich, weil die Organisation den öffentlichen Zugriff aus Datenschutz- oder anderen Gründen sperrt.

Das Personensuchsystem Whois (RFC 812, RFC 954) ermöglicht die Suche nach Personen (vor allem Netzwerkverwalter), die die folgenden Felder enthalten (im Suchergebnis):

- Vor- und Nachname
- Titel, akademischer Grad
- Organisation
- Abteilung
- Aufgabengebiet
- Kommentar
- Kommunikationsadressen: postalisch (dienstlich, privat), E-Mail, Telefon (dienstlich, privat), Fax
- Zeitpunkt der letzten Änderung des Eintrags

Nachteil bei Whois ist, daß der Zugriff zentral und nicht verteilt erfolgt. Rechner- und Netzwerkinformation kann nur über jeweils eine Organisation recherchiert werden, die die Daten zentral verwaltet (z.B. der europäische oder der amerikanische Whois-Service). Der Bestand ist daher auf die Zentrale beschränkt. Die Suche nach Whois-Rechnern ist nicht möglich. Weltweite Zentrale für Whois ist der Rechner "whois.internic.net".
Eine Weiterentwicklung von Whois ist das System Whois++ (RFC 1714, RFC 1835, RFC 1913), das einen verteilten Zugriff auf Personeninformation bietet.

Das Personeninformationssystem X.500 (ISO 9594) ist hierarchisch aufgebaut. Es bietet einen verteilten Zugriff auf Personeninformation. Als Suchmittel existieren die Attribute: Ländername, Name der Organisation, Name der organisatorischen Einheit und Personenname. Für das WWW existieren Benutzeroberflächen wie z.B.: X.500-WWW (1997).

Das Personeninformationssystem Netfind (Pu, Schwartz (1994)) faßt die Systeme Rusers, Rwho, Finger, Whois, X.500 u.a. zusammen und bietet eine übergreifende Suche nach Personeninformation. Für das WWW existieren Oberflächen (Netfind (1997)).

Häufig genutzte Web-Suchdienste sind E-Mail-Verzeichnisse, Telefonverzeichnisse, WWW-Adressenverzeichnisse (Homepage-Verzeichnisse), Telefonbücher und Branchenfernsprechbücher. Diese Dienste sind zumeist auf Regionen- und Ländergrenzen beschränkt. So existiert das vollständige amerikanische Telefon-Verzeichnis (Four11 (1997)), das deutsche Telefonbuch (DeTeMedien 2001a) oder das deutsche Branchenverzeichnis (DeTeMedien (2001b) aber noch kein vollständiges globales Telefon- oder Branchenverzeichnis. Umfangreiche globale Verzeichnisse finden sich für E-Mail-Adressen (Bigfoot (2001), Four11 (1997), IAF (2001)). In den Web-Suchdiensten werden als Suchmittel zumeist nur wenige Attribute angeboten. Die häufigsten sind: Name, Wohnort/Sitz, E-Mail-Adresse. Für kommerzielle Anwender werden bei Four11 (1997) weiterhin die Attribute Homepage-URL und Authentifizierungsschlüssel bereitgestellt.

Eine umfangreiche Auflistung von Internetsuchdiensten für Personen bietet das Hochschulbibliothekszentrum Nordrhein-Westfalen (HBZ (1997c)).

Für die Suche nach Gruppen (Firmen, Organisationen etc.) werden verschiedene Suchdienste im WWW angeboten (Apollo (1997), DIB (2001), LinkStar (1997), Switchboard (2001)). Bestand ist ein einzelnes Land oder die gesamte Erde. Als Suchmittel können die Attribute Geschäftsbereiche, Länder und Themen verwendet werden.

Eine Suche nach Mailinglisten wird mit Liszt (2000) ermöglicht. Suchmittel ist der Name oder ein Teil des Namens der Mailingliste. Weiterhin wird als Suchmittel eine Sachgebiets-hierarchie angeboten. Folgende Felder werden bereitgestellt: Name der Mailingliste, Schlag-wort(e), Beschreibung mit Informationsadresse und Kontaktperson (postalische Adresse, E-Mail-Adresse etc.).

Zusammenfassend ergeben sich folgende Felder:

Person:
- Name: Nachname, Vorname
- Titel, akademischer Grad
- Wohnorte: Straße, Hausnummer, Postleitzahl, Land
- Private und dienstliche Kommunikationsadressen: Postzustelladresse, Telefonnummer, Faxnummer, Telex, E-Mail-Adresse, Domainname
- Authentifizierungsschlüssel
- Inhaltliche Schlagworte
- Information über bearbeitete Projekte
- Persönliche Daten (Hobbies, Telefonnummer etc.)
- Informationsadressen: BTX, WWW, Hyper-G
- Typ: natürlich, virtuell, juristisch
- Zeitpunkt der letzten Änderung des Eintrags

Gruppe:
- Name
- Sitze: Erstsitz, Zweitsitz etc.: Straße, Hausnummer, Postleitzahl, Land
- Kommunikationsadressen (Postzustelladresse, Telefonnummer, Faxnummer, Telex, E-Mail-Adresse, Domainname
- Informationsadressen: BTX, WWW, Hyper-G
- Geschäftsbereiche, Projekte
- Inhaltliche Schlagworte
- Typ: Firma, Institution, Organisation, Verein, juristische Person, virtuelle Gruppe
- Mitglieder: eine Anzahl von Personen oder wieder Gruppen
- Zeitpunkt der letzten Änderung des Eintrags

Beispielanfragen

1. Welche Person hat eine Benutzerkennung auf dem Rechner "snake.cs.tu-berlin.de" und enthält im Namen die Zeichenfolge "Jo" ?
⇒ finger -l jo@snake.cs.tu-berlin.de

2. Welche Person hat eine Benutzerkennung auf dem Rechner "snake.cs.tu-berlin.de" und hat exakt die Benutzerkennung "erhard" ?
⇒ finger -lm erhard@snake.cs.tu-berlin.de

3. Welche Personen sind auf dem Rechner "snake.cs.tu-berlin.de" eingelogt ?
⇒ finger @snake.cs.tu-berlin.de

4. Welche Personen der Organisation "TH-Darmstadt" haben den Vornamen "Andreas"
⇒ whois -h whois.th-darmstadt.de Andreas

5. Welche Personen haben den Nachnamen "Reiner" und den Vornamen "Ulrike" ?
Mit Bigfoot (2001) können mit der "advanced search" die betreffenden Felder eingegeben werden

6. Welche Firmen haben den Geschäftsbereich "Computers Dealers" und haben ihren Sitz in der Stadt "New York" im Staate "NY" ?
Mit Switchboard (2001) kann mit der "Business-Search" eine entsprechende Suche durchgeführt werden.

3.1.2 Rechnersuche

Das Informationsssystem Whois (RFC 812, RFC 954) bzw. Whois++ (RFC 1714, RFC 1913) ermöglicht die Suche nach Rechnern und Netzwerken. Suchmittel sind IP-Adressen oder Teile von IP-Adressen. Es werden folgende Felder bereitgestellt:

Rechner:

- Domain-Name
- IP-Nummer
- Systemadministrator: Name, Telefon, postalische Adresse, E-Mail-Adresse
- Rechnertyp
- Betriebssystem
- Zeitpunkt der letzten Modifikation des Eintrags
- FTP-Adresse mit weiterer Information

Netzwerk:

- Domain-Name
- Administrator: Name, Telefon, postalische Adresse, E-Mail-Adresse
- Technische Kontaktperson: Name, Telefon, postalische Adresse, E-Mail-Adresse
- Namen der Nameserver des Netzwerks
- Zeitpunkt der letzten Modifikation des Eintrags

Der Nachteil bei Whois besteht darin, daß der Zugriff zentral und nicht verteilt erfolgt. Rechner- und Netzwerkinformation kann nur über jeweils eine Organisation recherchiert werden, die die Daten zentral verwaltet (z.B. der europäische oder der amerikanische Whois-Service). Der Bestand ist daher auf die Zentrale beschränkt. Die Suche nach Whois-Rechnern ist nicht möglich. Weltweite Zentrale für Whois ist der Rechner "whois.internic.net".
Eine Weiterentwicklung von Whois ist das verteilte System Whois++ (RFC 1714, RFC 1835, RFC 1913).

Netfind (Pu, Schwartz (1994)) ermöglicht die Suche nach Rechnern. Es wird eine einfache Volltextsuche mit allen Attributen durchgeführt. Für Netfind existiert eine einfache WWW-Benutzeroberfläche (Netfind (1997)).

Archie (s.u.) bietet eine Suche nach Archie- und FTP-Servern und nach Rechnerdomainnamen.

Chemie (1997) bietet eine Suche nach WWW-Servern mit einer WWW-Benutzeroberfläche. Suchmittel ist ein regulärer Ausdruck für alle Attribute. Folgende Felder werden unterschieden: Stadt, Name des Servers, Land, WWW-Adresse.

Beispielanfragen

1. Welche Rechner-Domainnamen existieren ?
⇒ telnet archie.th-darmstadt.de
⇒ domains

2. Welches Netzwerk hat die IP-Adresse "130.149" ?
⇒ whois -h whois.internic.net 130.149

3. Welches Netzwerk hat die IP-Adresse "129.17" ?
⇒ whois -h whois.internic.net 129.17

4. Welche Rechner enthalten in einem ihrer Attributwerte die Zeichenfolge "snake" ?
⇒ telnet ds.internic.net
⇒ netfind
⇒ 3
⇒ 2
⇒ snake

5. Welche Archie-Server existieren ?
⇒ telnet archie.th-darmstadt.de
⇒ servers

6. Welche FTP-Server benutzt Archie für den Aufbau seiner Datenbank
⇒ telnet archie.th-darmstadt.de
⇒ list

3.1.3 Dateisuche

Dateiinformationssysteme gehören zu den am häufigsten benutzten Systemen im Internet. Wir untersuchen die wichtigsten Dateisysteme und die Systeme Telnet, FTP, Alex und Archie.

3.1.3.1 Dateisysteme

Dateisysteme sind i.A. monohierarchisch und azyklisch aufgebaut. Man kann sie deshalb auch als gerichtete azyklische Graphen betrachten, deren Knoten den Dateien im Dateisystem entsprechen und deren unbenannte Kanten die Hierarchie der Dateiverzeichnisse wiederspiegeln. Durch Dateiverweise kann das Dateisystem zyklisch werden.
Dateien können in Formate eingeteilt werden. Man unterscheidet zum Beispiel: Textdatei, ausführbare Datei, Graphikdatei, Videodatei, gepackte Datei. Weitere werden mit dem Mime-Standard (RFC 1521, RFC 1522) definiert.
Dateien besitzen eine feste Menge von Feldern wie zum Beispiel: Id, Inhalt, Name, Verzeichnis, Erstellungsdatum und- Uhrzeit, Größe in Bytes, Benutzer/Gruppenkennung und Benutzerrechte.
Neuerdings werden Dateisysteme entwickelt, die die Modifikation der Felder erlauben (Khalidi et.al. (1993)).

Die Feldwerte sind abhängig vom verwendeten Zeichensatz und der Feldlänge.
Einen Überblick über Dateisysteme geben Tanenbaum (1995), Gulbins, Obermayer (1995) und Coulouris, Dollimore (1994).

3.1.3.1.1 Häufig verwendete Dateisysteme

Betriebssystem	Dateisysteme	Literaturquellen
DOS, Windows 3.1.x (Microsoft)	File Allocation Table (FAT), Compact Disk System	DOS (1994), Windows (1992)
OS/2 (IBM)	FAT, Super FAT, High Performance File System (HPFS), Compact Disk System	Huttel (1995)
MacOS, A/UX (Apple)	Hierarchical File System (HFS)	Apple (1992)
Novell Netware	Netware File System (Volume Entry Tables, Directory Entry Tables, FAT, Super FAT), Compact Disk File System (HSFS nach ISO 9660), weitere Dateisysteme durch "names spaces": HPFS, MacFS und network file system (nfs nach RFC 1094)	Lauer, Scholz (1996)
Windows NT (Microsoft)	New Technology File System (NTFS), FAT, HPFS, Netware File System, Compact Disk System (HSFS nach ISO 9660)	Custer (1993)
AIX (IBM), HP-UX (HP), Sinix (Siemens), Solaris/Sun-OS (SUN)	system V file system (s5), unix file system (ufs), boot file system (bfs), Compact Disk Systeme (HSFS nach ISO 9660), network file system (nfs), remote file system (rfs)	Handschuch (1995), Gulbins, Obermayr (1995), Stern (1995)
Unixware (Novell)	veritas file system (vxfs)	
Linux	extended file system (ext), FAT, eingeschränkt HPFS (read only), Compact Disk Systeme (HSFS nach ISO 9660), nfs	Kofler (1995)
Ultrix, OSF/1 (DEC)	unix file system (ufs), network file system (nfs)	

Tabelle 4: **Häufig verwendete Dateisysteme**

Momentan werden Anstrengungen unternommen, Dateisysteme für das WWW bereitzustellen. Beispiele sind WebNFS (WebNFS (2001)), CIFS (CIFS (1997)) und IFS (IFS (2000)).

3.1.3.1.2 Physikalische und logische Sicht

Für den Zugriff auf Daten sind folgende Ebenen zu unterscheiden:
1. Physikalische Ebene
2. Logische Ebene
3. Benutzer/Präsentationsebene

Auf der physikalischen Ebene werden Anfragen in Abhängigkeit von den verwendeten Datenstrukturen (z.B. auf physikalischer Dateiebene) gestellt.

Auf der logischen Ebene wird physikalische Datenunabhängigkeit hergestellt. Der Zugriff auf Daten abstrahiert von der Implementation. Dazu werden Daten in Datentypen mit dazugehörigen Feldmengen eingeteilt.

Auf der Benutzer- bzw. Präsentationsebene wird eine logische Datenunabhängigkeit hergestellt. Der Zugriff auf die Daten abstrahiert von den logisch definierten Datenbeschreibungen und gestattet einen einheitlichen Zugriff auf mehrere logisch unterschiedliche Datenbeschreibungen.

Wir wollen für Dateisysteme die Trennung der physikalischen von der logischen Sicht genauer betrachten:

Physikalische Sicht: Datenbestände werden mit Hilfe des Dateisystems auf Datenträgern (Festplatten, Disketten, CD-Rom-Laufwerke, Bandlaufwerke etc.) gespeichert. Datenträger werden in ein oder mehrere Teile (Partitionen) aufgeteilt. Ein Volume bezeichnet ein oder mehrere Partitionen (statt Volume werden auch die Bezeichnungen: Laufwerksbuchstaben, Gerätedateien etc. benutzt). Volumes bestehen aus Dateien (Dateiverzeichnisse, normale Dateien, Dateiverweise etc.). Dateiverzeichnisse bestehen aus Dateien.

Logische Sicht: Ein logisches Datenverzeichnis ist die hierarchische Zusammenfassung von Dateibeständen im Dateisystem ohne Berücksichtigung des Datenträgers, auf dem die Dateien abgelegt werden. Es besteht aus Dateien unterschiedlichen Typs (Dateiverzeichnisse, normale Dateien, Dateiverweise etc.). Dateien vom Typ Dateiverzeichnis bestehen aus Dateien. Eine Datei kann Bestandteil mehrerer Dateiverzeichnisse sein.

In den Dateisystemen werden logische und physikalische Ebene unterschiedlich stark getrennt:

FAT, HPFS: schwache Trennung: Der Gesamtdatenbestand wird physikalisch auf genau einem Rechner mit ein oder mehreren Datenträgern gespeichert. Laufwerksbuchstaben (a:, b:, c:, d:,...,z:) bezeichnen jeweils genau eine Partition eines Datenträgers. Ein Laufwerksbuchstabe besteht aus Dateien (Dateiverzeichnisse, normale Dateien).

HFS: schwache Trennung: Der Gesamtdatenbestand wird physikalisch verteilt auf verschiedenen Rechnern mit mehreren Datenträgern gespeichert. Ein Volume bezeichnet genau eine Partition eines Datenträgers. Ein Volume besteht aus Dateien (Dateiverzeichnisse, normale Dateien). Der Dateiinhalt besteht nicht wie in den anderen Dateisystemen aus einer Folge von Bytes, sondern ist in zwei Teile getrennt: Data-Fork und Resource-Fork. Data-Fork besteht aus einer Folge von Bytes, der Resource-Fork enthält strukturierte Daten wie Menüs, Dialogboxen oder ausführbare Programme.

Netware-FS: schwache Trennung: Der Gesamtdatenbestand wird physikalisch verteilt auf verschiedenen Rechnern mit mehreren Datenträgern gespeichert. Ein Volume bezeichnet ein oder mehrere Partitionen auf den Datenträgern (ein Volume kann über mehrere Datenträger gebildet werden). Ein Volume besteht aus Dateien (Dateiverzeichnisse, normale Dateien).

NTFS: schwache Trennung: Der Gesamtdatenbestand wird physikalisch verteilt auf verschiedenen Rechnern mit mehreren Datenträgern gespeichert. Ein Volume bezeichnet ein oder mehrere Partitionen eines Datenträgers; ein Laufwerksbuchstabe (a:, b:, c:, d:,...,z:) bezeichnet

genau eine Partition eines Datenträgers. Ein Volume oder Laufwerksbuchstabe besteht aus Dateien (Dateiverzeichnisse, normale Dateien).
ufs/nfs: starke Trennung: Der Gesamtdatenbestand wird physikalisch verteilt auf verschiedenen Rechnern mit mehreren Datenträgern gespeichert. Das logische Gesamtverzeichnis besteht aus Dateien (Dateiverzeichnisse, normale Dateien, Gerätedateien etc.).

3.1.3.1.3 Felder von Dateien

Wir nennen im Folgenden Felder von Dateien, die auf der logischen Ebene für Informationssuchende bei der Dateisuche einen Informationsgehalt versprechen.

FAT: Inhalt, Name, Verzeichnisname (wenn Datei), letzter Modifikationszeitpunkt des Inhalts der Datei (Datum, Uhrzeit), Version, Copyright, Größe in Bytes und weitere Felder (unsichtbar, schreibgeschützt, archiviert, Systemdatei)
HPFS: Felder wie bei FAT, zugeordnete Ikone, Name der zugeordneten Ikone, Dateityp, Erstellungszeitpunkt (Datum, Uhrzeit), letzter Zugriffszeitpunkt (Datum, Uhrzeit), Deskriptor, Protokoll, Kommentar, weitere vom Benutzer festgelegte Felder (Feldnamen bestehen aus beliebigen Zeichen des Zeichensatzes und können bis zu 255 Zeichen lang sein; insgesamt sind alle Felder nicht größer als 64 KB)
HFS: Dateiinhalt, Resourceinhalt (Menü, Ikone, Fenstergröße, ausführbarer Code etc.), Name, Verzeichnisname (wenn Datei), Zeitpunkt des Anlegens der Datei (Datum, Uhrzeit), letzter Modifikationszeitpunkt des Inhalts der Datei (Datum, Uhrzeit), letzer Backup-Zeitpunkt der Datei, Version, Größe in Bytes, Position der Dateimarke, weitere Dateiattribute (gesperrt, Dateninhalt geöffnet, Resourceninhalt geöffnet), Benutzerkennung, Gruppenkennung, Benutzer/Gruppenrechte (lesen, ausführen, schreiben),
Netware-FS: Felder wie bei FAT, Rechnername, Volumename, Modifizieren, CI (Macintosh Copy Inhibit: nur Datei), DC (Don't compress), DI (Delete inhibit), DM (Don't migrate), DS (Don't Suballocate: nur Datei), H (Hidden), I (Index: nur Datei), IC (Immediate Compress, N (Normal: lesen, schreiben und mehrfach benutzbar), P (Purge), RI (Rename Inihibit), SH (Shareable: nur Datei), T (Transactional: nur Datei), X (Execute only: nur Datei), Benutzer/Gruppenrechte (lesen/suchen, ausführen, erstellen, löschen, umbenennen), Benutzer/Gruppenrechte für Eigenschaften von Einträgen (vergleichen, lesen, schreiben, eigenen Namen hinzufügen oder löschen).
NTFS: Felder wie bei HPFS, verknüpfte Anwendung, Benutzerkennung, Gruppenkennung, Benutzer/Gruppenrechte (anzeigen, hinzufügen, lesen, hinzufügen und lesen, schreiben, ausführen, löschen, Benutzerrechte ändern, Besitz übernehmen), Name des Datenträgers, Parameter des Datenträgers.
ufs/nfs: Identifikationsnummer (i-node), Inhalt, Name, Verzeichnis (wenn Datei), letzter Modifikationszeitpunkt des Inhalts der Datei (Datum, Uhrzeit), letzter Modifikationszeitpunkt der Dateiattribute (Datum, Uhrzeit), letzter Zugriffszeitpunkt (Datum, Uhrzeit), Größe in Bytes, Benutzerkennung (Identifikationsnummer, Name), Gruppenkennung (Identifikationsnummer, Name) und Benutzer/Gruppenrechte (lesen, schreiben, ausführen), Dateityp (Datei, Verzeichnis, Gerätedatei), Anzahl der Links, Anzahl der assoziierten Speicherblöcke.

Werte des Feldes "Dateiname"

FAT: Der Dateiname (Dateiverzeichnis, normale Datei) besteht aus Zeichen des Zeichensatzes (128 verschiedene Zeichen durch 7-Bit nach ISO 646) außer den Zeichen ". , ; " / \ : [ ] = | Leerzeichen". Ein Dateiname kann bis zu 8 Zeichen lang sein gefolgt von einem "." und bis zu 3 weiteren Zeichen. Groß/Kleinschreibung wird nicht unterschieden.

Durch den DOS-Befehl "chcp" kann der Zeichensatz erweitert werden (8-Bit nach ISO 8859-X; es werden jedoch nicht alle Zeichensätze der Norm unterstützt).
HPFS: Der Dateiname (Datei, Verzeichnis) besteht aus Zeichen des Zeichensatzes (256 verschiedene Zeichen durch 8-Bit nach ISO 8859) außer den Zeichen: "\ / : * ? " < > | & + -". Der Dateiname kann bis zu 254 Zeichen lang sein. Groß/Kleinschreibung wird unterschieden.
HFS: Der Dateiname besteht aus den Zeichen des Zeichensatzes (8-Bit nach ISO 8859) außer dem Zeichen ":" Ein Dateiname kann bis zu 31 Zeichen, ein Volumename bis zu 27 Zeichen lang sein. Dateien dürfen nicht mit einem Punkt beginnen. Groß/Kleinschreibung wird zwar unterschieden, allerdings ist die Großschreibung nicht signifikant. Das heißt, daß in einem Dateiverzeichnis nie zwei Dateien existieren, die sich im Dateinamen nur hinsichtlich der Groß/Kleinschreibung unterscheiden.
Netware-FS: Namen (Rechner, Volume, Verzeichnis, Datei) können bis zu 15 Zeichen lang sein und bestehen aus Zeichen des Zeichensatzes (Großbuchstaben: A-Z, Ziffern: 0-9 und Unterstriche). Für ein Volume können zusätzlich weitere Namenskonventionen ("name spaces") für Dateien festgelegt werden. Unterstützt werden HPFS, HFS, und nfs mit den jeweils geltenden Zeichensätzen und jeweiligen Größenangaben für die Attributwerte.
NTFS: Der Dateiname (Datei, Verzeichnis) besteht aus den Zeichen des Alphabets Unicode (65536 verschiedene Zeichen durch 16-Bit nach ISO 10646) außer den Zeichen "? " / \ < > * | : Leerzeichen". Der Name kann bis zu 256 Zeichen lang sein. Groß/Kleinschreibung wird unterschieden.
ufs/nfs: Der Name eines Eintrags (Datei, Verzeichnis) besteht aus Zeichen des Alphabets (256 verschiedene Zeichen durch 8-Bit nach ISO 8859-X) außer dem Zeichen "/" . Das Leerzeichen kann zwar verwendet werden, ist jedoch mit Vorsicht zu genießen. Der Name kann bis zu 255 Zeichen lang sein. Groß/Kleinschreibung wird unterschieden.

Dateiverweise

FAT, HPFS, **HFS**, **Netware:** Nicht möglich
NTFS: Es können Verweise zwischen den Einträgen (Hard-Links) gebildet werden.
ufs/nfs: Es können Verweise zwischen den Einträgen (Hard-Links) und Verweise zwischen den Eintragsnamen (symbolische Links) gebildet werden.

3.1.3.1.4 Anfragen nach Dateien

Die Dateisysteme unterscheiden sich hinsichtlich der möglichen Anfragen und ihren Ergebnissen deutlich. Während z.B. unter dem Betriebssystem UNIX eine Anfrage nach Dateien mit dem Attribut Benutzerkennung gestellt werden kann, existiert unter DOS kein Feld Benutzerkennung, so daß diese Anfrage dort nicht möglich ist. Auch die Semantik von Anfragen divergiert in den unterschiedlichen Betriebs- und Dateisystemen: Eine Anfrage nach Dateien mit dem Dateinamen "b[au]g.*" als Menüanfrage unter dem Dateimanager unter MS-Windows liefert die Menge von Dateien, die mit der Zeichenfolge "b[au]g" beginnen. Dieselbe Anfrage unter UNIX mit dem Kommando find "b[au]g.*" liefert alle Dateinamen, die mit dem Zeichen "b" beginnen, gefolgt von einem der beiden Zeichen "a" oder "u", gefolgt von dem Zeichen "g", gefolgt von dem Zeichen ".", gefolgt von einer beliebigen Zeichenfolge.

Anfragekommandos
DOS: attrib, dir, find , tree
OS-2: Anfragen wie bei DOS, Menüsuche nach Dateien

Novell-Netware: Menüsuche nach Dateien, Netware unterstützt die Anbindung der Betriebssysteme DOS, OS/2, MacOS, Windows NT und Unix. Für das Netware-FS können die Anfragekommandos dieser Betriebssysteme verwendet werden.
Windows NT: Anfragen wie bei DOS, Menüsuche nach Dateien
Unix: dir, egrep, file, find, grep, locate, ls, pwd, stat

Ordnung der Anfrageergebnisse
DOS:
- **dir**
  nach dem Namen: in (umgekehrter) alphabetischer Reihenfolge
  nach der Dateierweiterung: in (umgekehrter) alphabetischer Reihenfolge
  nach dem letzten Modifikationszeitpunkt: in (umgekehrter) zeitlicher Reihenfolge
  nach der Größe: (umgekehrt) nach Größe der Datei
  nach Dateityp: Verzeichnisse hinter/vor Dateien gruppiert
  nach Komprimierung: (umgekehrt) nach Komprimierungsverhältnis geordnet

OS/2:
- **Menüsuche**
  nach jedem Attribut (beispielsweise nach dem Modifikationszeitpunkt)

Novell Netware:
  siehe DOS, OS/2, MacOS, Windows NT und Unix

Windows NT:
- **Menüsuche**
  nach dem Namen: in alphabetischer Reihenfolge
  nach der Dateierweiterung: in alphabetischer Reihenfolge
  nach dem letzten Modifikationszeitpunkt: in zeitlicher Reihenfolge
  nach der Größe: Reihenfolge nach Größe der Datei

UNIX:
- **ls**
  nach dem Namen: in (umgekehrter) alphabetischer Reihenfolge
  nach der Dateierweiterung: in (umgekehrter) alphabetischer Reihenfolge
  nach dem letzten Modifikationszeitpunkt: in (umgekehrter) zeitlicher Reihenfolge
  nach der Größe: Reihenfolge (umgekehrt) nach Größe der Datei

Reguläre Ausdrücke
DOS, OS-2, MacOS, Novell Netware und Windows NT bieten keine Suchmittel für reguläre Ausdrücke. Bestimmte Programme wie z.B. Norton Commander bilden diese Möglichkeit jedoch nach.
Unix bietet für reguläre Ausdrücke die Anwendungsprogramme find (stark eingeschränkte reguläre Ausdrücke), grep (eingeschränkte reguläre Ausdrücke), agrep (eingeschränkte reguläre Ausdrücke bis auf Zeichenwiederholung), egrep (vollständige reguläre Ausdrücke) und ed, ex, vi (erweiterte reguläre Ausdrücke).

Beispielanfragen
Da UNIX über das vielfältigste Angebot von Anfragekommandos für unsere Zwecke verfügt, betrachten wir diese hier genauer. Einige Anfragen lassen sich auch anders als hier angegeben bilden.

Anfragen nach Dateien

1. Welche Dateinamen vom Typ Dateiverzeichnis befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" oder in einem Unterverzeichnis von "/usr" ?
⇒ telnet snake.cs.tu-berlin.de
⇒ find /usr/* -type d

Terminal - snake.cs.tu-berlin.de
File Edit Session Options Help

```
snake josefw 3 (~):
snake josefw 3 (~): find /usr/* -type d
/usr/4lib
/usr/TT_DB
/usr/X11
/usr/X11/R5
/usr/X11/R5/lib
/usr/X11/R5/lib/nls
/usr/X11/R5/lib/nls/local_im_tbl
/usr/X11/bin
/usr/X11/include
/usr/X11/include/X11
/usr/X11/include/X11/bitmaps
/usr/X11/include/X11/extensions
/usr/X11/include/X11/fonts
/usr/X11/include/X11/ICE
/usr/X11/include/X11/SM
/usr/X11/include/X11/Xmu
/usr/X11/include/X11/Xaw
/usr/X11/include/X11/PEX5
/usr/X11/include/X11/Xaw3d
/usr/X11/lib
/usr/X11/lib/app-defaults
/usr/X11/lib/config
/usr/X11/lib/fonts
/usr/X11/lib/fonts/misc
/usr/X11/lib/fonts/75dpi
/usr/X11/lib/fonts/100dpi
^C
snake josefw 4 (~):
```

Abbildung 2: find /usr/* -type d (im März 1997)

2. Welche Dateinamen befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr", deren Inhalt die Zeichenkette "file system" enthält. Groß/Kleinschreibung wird nicht unterschieden?
⇒ telnet snake.cs.tu-berlin.de
⇒ grep -il 'file system' /usr/*

3. Welche Dateinamen befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" oder in einem Unterverzeichnis von "/usr", deren Inhalt die Zeichenkette "file system" enthält ?
⇒ telnet snake.cs.tu-berlin.de
⇒ grep -l 'file system' `find /usr -print`

4. Welche Textzeilen auf dem Rechner "snake.cs.tu-berlin.de" aus dem Inhalt von Dateien im Verzeichnis "/usr" oder in einem Unterverzeichnis von "/usr" enthalten die Zeichenfolge "file system" ?
⇒ telnet snake.cs.tu-berlin.de
⇒ grep 'file system' `find /usr/* -print`

5. Welche Dateinamen befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" oder in einem Verzeichnis bis zur Tiefe 3 unterhalb von "usr"?
Diese Anfrage ist direkt unter UNIX nicht möglich, kann jedoch z.B. mit einfachen Shell-Befehlen programmiert werden.

Boolesche Anfragen

6. Welche Dateinamen befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" oder in einem Verzeichnis unterhalb von "/usr", deren Benutzerkennung "laszlo" heißt? Das Ergebnis soll alphabetisch nach den Namen geordnet werden.

⇒ telnet snake.cs.tu-berlin.de

⇒ cd /usr, c) ls -laR | grep laszlo

Bei dieser Anfrage werden Einträge als Ergebnis geliefert, die in irgendeinem Attributwert die Zeichenfolge "laszlo" enthalten.

Besser ist find geeignet:

⇒ telnet snake.cs.tu-berlin.de

⇒ find /usr -user laszlo -ls

7. Welche Dateinamen befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" oder in einem Unterverzeichnis von "/usr", die vom Typ Dateiverzeichnis sind oder deren Benutzerkennung "laszlo" heißt ?

⇒ telnet snake.cs.tu-berlin.de

⇒ find /usr -type d -o -user laszlo -ls

8. Welche Dateinamen befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" oder in einem Unterverzeichnis von "/usr", die vom Typ Dateiverzeichnis sind und deren Benutzerkennung "laszlo" heißt ?

⇒ telnet snake.cs.tu-berlin.de

⇒ find /usr -type d -a -user laszlo -ls

9. Welche Dateinamen befinden sich auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" oder in einem Unterverzeichnis von "/usr" oder in mit den Unterverzeichnissen durch symbolischen Link in Beziehung stehenden Verzeichnissen, die vom Typ Dateiverzeichnis sind und deren Gruppenkennung "wbs" heißt ?

⇒ telnet snake.cs.tu-berlin.de

⇒ find /usr -type d -follow -a -group wbs -ls

Anfragen mit regulären Ausdrücken

10. Welche Dateinamen auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/usr" enthalten in ihrem Inhalt den regulären Ausdruck "f.*m" (Textstellen, die mit dem Zeichen "f" beginnen, gefolgt von einer beliebigen Zeichenfolge und die mit dem Zeichen "m" abschließen) ?

⇒ telnet snake.cs.tu-berlin.de

⇒ grep -l 'f.*m' /usr/*

11. Welche Textstellen auf dem Rechner "snake.cs.tu-berlin.de" im Verzeichnis "/home/wbs/josefw/test" aus dem Inhalt der Datei "a" enthalten exakt die Zeichenfolge "hallo" (exakt heißt, daß vor und nach der gesuchten Zeichenfolge kein Zeichen aus der Menge der Worttrennzeichen (Leerzeichen, Tabulator, "." "," "_" etc.)vorkommt)?

⇒ telnet snake.cs.tu-berlin.de

⇒ cd /home/wbs/josefw/test

⇒ vi a

⇒ /\<hallo\>/ (wird für jedes Vorkommen der gesuchten Textstelle eingegeben)

12. Wieviele Textzeilen auf dem Rechner "snake.cs.tu-berlin.de" im Inhalt aller Dateien im Verzeichnis "/home/wbs/josefw/test" enthalten den regulären Ausdruck "[Ff]ile ?[Ss]ystem?" (Textzeilen, die mit dem Zeichen "F" oder "f" beginnen, gefolgt von der Zeichenfolge "ile", gefolgt von 0 bis 1 beliebigen Zeichen, gefolgt von den Zeichen "S" oder "s", gefolgt von der Zeichenfolge "ystem", gefolgt von 0 bis 1 beliebigen Zeichen) ?
⇒ telnet snake.cs.tu-berlin.de
⇒ egrep -c '[Ff]ile ?[Ss]ystem?' *

13. Welche Textzeilen auf dem Rechner "snake.cs.tu-berlin.de" im Inhalt aller Dateien im Verzeichnis "/home/wbs/josefw/test" enthalten Zeichenfolgen beginnend mit 2 beliebigen Zahlen, gefolgt vom Zeichen ".", gefolgt von 2 beliebigen Zahlen, gefolgt vom Zeichen ".", gefolgt von 4 beliebigen Zahlen (ein Datumsformat) ?
⇒ telnet snake.cs.tu-berlin.de
⇒ grep '[0-9]\{2\}.[0-9]\{2\}.[0-9]\{4\}' *

14. Welche Textzeilen auf dem Rechner "snake.cs.tu-berlin.de" im Inhalt aller Dateien im Verzeichnis "/home/wbs/josefw/test" enthalten Zeichenfolgen, die eine Länge von mehr als 20 Zeichen haben ?
⇒ telnet snake.cs.tu-berlin.de
⇒ grep \.\\\{20\,\\\} * (das Zeichen "\" ist das Fluchtzeichen für die Metazeichen)

15. Welche Textzeilen auf dem Rechner "snake.cs.tu-berlin.de" im Inhalt aller Dateien im Verzeichnis "/home/wbs/josefw/test" enthalten Zeichenfolgen, die eine Länge von weniger als "20" Zeichen haben ?
Diese Anfrage ist als regulärer Ausdruck nicht möglich, kann aber einfach programmiert werden.

16. Welche Textzeilen auf dem Rechner "snake.cs.tu-berlin.de" im Inhalt aller Dateien im Verzeichnis "/home/wbs/josefw/test" enthalten den regulären Ausdruck "compan(y|ies)" (Zeichenfolgen beginnend mit der Zeichenfolge "compan", gefolgt von dem Zeichen "y" oder der Zeichenfolge "ies") ?
⇒ telnet snake.cs.tu-berlin.de
⇒ egrep compan\(y\|ies\) *

17. Welche Textzeilen auf dem Rechner "snake.cs.tu-berlin.de" aus dem Inhalt von Einträgen im Verzeichnis "/usr" oder in einem Unterverzeichnis von "/usr" enthalten die Zeichenfolge "file system" oder "file systems" oder "filesystem" oder "filesystems"?
⇒ telnet snake.cs.tu-berlin.de
⇒ egrep 'file system|file systems|filesystem|filesystems' `find /usr/* -print`

Leistungsgrenzen

- Kein universelles und standardisiertes Dateisystem
- Unterschiedliche Zeichensätze in den jeweiligen Dateisystemen
- Unterschiedliche Attributmengen in den jeweiligen Dateisystemen
- Spezifizierung von Attributen ist bei der Suche nur teilweise möglich
- Dateibestand ist auf den Datenträger bzw. die logische Ebene des Dateisystems beschränkt
- Keine einheitliche boolesche Verknüpfung möglich

3.1.3.2 Telnet

Terminal Emulation (Telnet) ist eine der frühesten Anwendungen/Protokolle im Internet. 1983 wurde die diesbezügliche Norm RFC 854 veröffentlicht. Für das Arbeiten mit Telnet ist i.A. eine Zugangsberechtigung (Benutzerkennung mit Passwort) erforderlich. Viele Rechner im Internet, insbesondere Rechner, die öffentlich zugängliche Information anbieten, erlauben einen Gast-Zugang oder verlangen keine Zugangskennung.

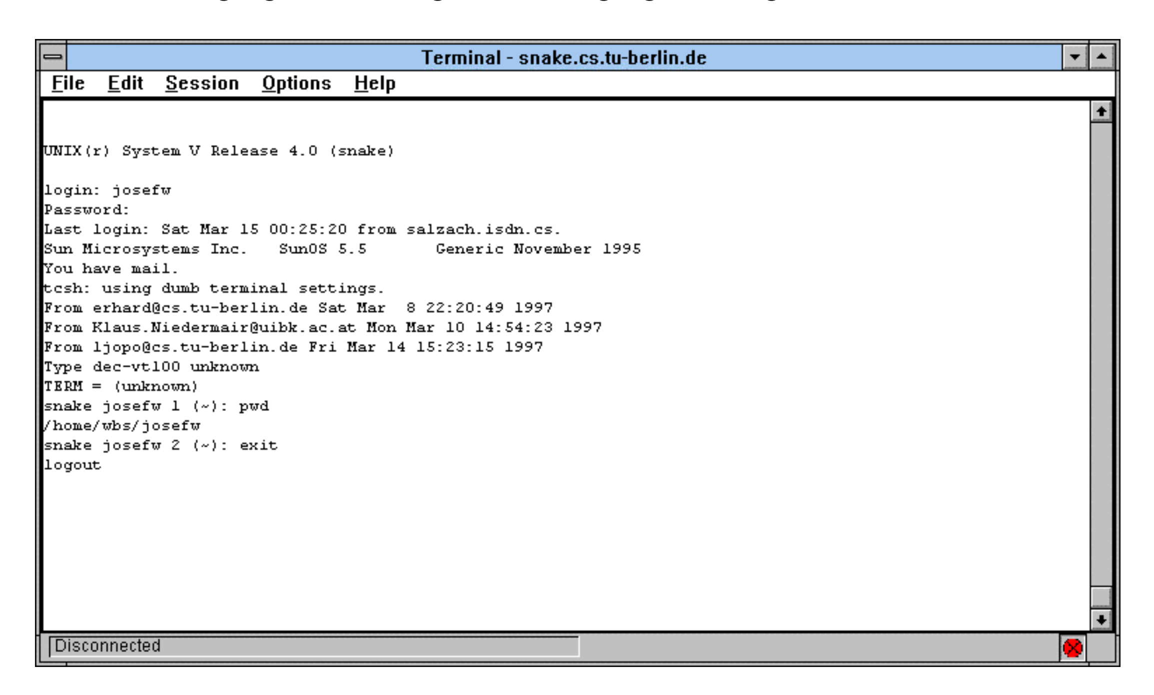


Abbildung 3: Eine einfache Telnetsitzung (im März 1997)

Mit Telnet können Anwendungssysteme automatisch aufgerufen werden. Beispielsweise wird bei einer Telnetverbindung zum Rechner "echo.lu" mit der Benutzerkennung "ECHO" das Anfragesystem Grips 6.0 automatisch gestartet.
Wenn ein Anwender sich auf einem entfernten Rechner eingewählt hat, kann er die Möglichkeiten des dort vorhandenen Betriebssystems und die dort angebotenen Anwendungssysteme (wie zum Beispiel Dateisysteme, Datenbanksysteme, Bibliothekssysteme, Gopher, WAIS etc.) nutzen. In Reiner (1991) wird für diese Zwecke eine universelle Anfragesprache für Informationssysteme entwickelt.
Mit Telnet kann nur auf einem Rechner gearbeitet werden, dessen Name (Domainname oder IP-Adresse) bekannt ist. Mit HyTelnet (Scott (1997), HyTelnet (1997)) wird eine inhaltliche Strukturierung der Telnet-Rechner gegeben. Weiterhin können mit HyTelnet Anwenderprogramme gestartet werden.

3.1.3.3 FTP

Die Dateiübertragung mit FTP (File Transfer Protocol) ist einer der am meisten genutzten Dienste im Internet. Durch FTP wird die Übertragung von Dateien zwischen Rechnern ermöglicht. Es ist möglich, sowohl "private" als auch "öffentliche" Dateien von einem Rechner zum anderen zu übertragen.
Mit Hilfe von FTP-Servern werden Einträge (Dateien und Verzeichnisse) öffentlich bereitgestellt. Mit den unterschiedlichen Anwendungssystemen (Clients) können Einträge manipuliert und übertragen werden: Speichern, Anfügen, Umbenennen, Verschieben, Anfordern.

Es existieren viele öffentlich zugängliche FTP-Informationssysteme. Information wird in Form von Dateien angeboten: Software, Handbücher, Informationstexte, häufig gestellte Fragen und Antworten, Texte aus Bereichen wie zum Beispiel Normen, wissenschaftliche Berichte und Romane und Dokumentformate wie Bilder, Tondokumente oder Videos.
FTP wird i.A. zwischen zwei Rechnern benutzt, die mit dem 8-Bit Zeichensatz ISO 8859 arbeiten.
Wenn FTP zwischen zwei Rechnern benutzt wird, die mit unterschiedlichen Zeichensätzen arbeiten, ist eine Konvertierung der übertragenen Daten nötig. Dies kann entweder manuell (beispielsweise mit dem Kommando ntrans) oder automatisch durchgeführt werden.
Weiterhin können Dateien, die in komprimierter Form vorliegen, bei der Übertragung automatisch entkomprimiert werden.
Man braucht, um mit FTP in nichtöffentlichen Dateiverzeichnissen arbeiten zu können, i.A. eine Benutzerkennung für zwei Rechner: 1. für den Rechner, von dem man FTP benutzt (FTP-Benutzerrechner) und 2. für den Rechner, auf den man per FTP zugreifen will (FTP-Ziel-Rechner). Um diesen Nachteil aufzuheben hat Horlacher (2001) eine asynchrone Dateiübertragung entwickelt. Ein RFC dafür ist in Vorbereitung.
Mit FTP ist es nicht möglich, mehrere FTP-Server in einer Anfrage zu behandeln. Systeme wie Archie, FTP-Search und Alex/Prospero ermöglichen Anfragen über mehrere FTP-Server.
Wichtig sind die Verfahrenschritte bei einem Zugriff auf FTP-Einträge. Als erster Schritt wird eine Verbindung zu einem FTP-Server aufgebaut. Danach können Anfragen mit den unterschiedlichen Anfragekommandos gestellt werden. Die wichtigsten sind:

- "!": Wechsel auf die Betriebssystemebene des lokalen Rechners durchgeführt.
- "cd": Wechsel in das angegebene Verzeichnis auf dem FTP-Rechner
- "cdup": Wechsel in das nächsthöhere Verzeichnis auf dem FTP-Rechner
- "dir": alphabetisch sortierte Ausgabe der Einträge des angegebenen Verzeichnisses (dieselbe Semantik wie Unix-Befehl "ls-lag")
- "lcd": Wechsel in das angegebene Verzeichnis auf dem lokalen Rechner
- "ls": alphabetisch sortierte Ausgabe der Namen der Einträge des angegebenen Verzeichnisses (ohne Parameter dieselbe Semantik wie der Unix-Befehl "ls")
- "modtime": Ausgabe des letzten Modifikationszeitpunkts der angegebenen Datei
- "ntrans": Konvertierungsfunktion für Zeichen (für die Dateiübertragung)
- "pwd": Ausgabe des aktuellen vollständigen Verzeichnisnamens auf dem FTP-Rechner
- "size": Ausgabe der Größe (in Bytes) des angegebenen Eintrags

Beispielanfragen

1. Welche Dateinamen befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub" ? Das Ergebnis soll alphabetisch nach den Namen geordnet werden.
⇒ ftp ftp.zrz.tu-berlin.de
⇒ cd pub
⇒ ls
oder mit einer WWW-Benutzeroberfläche:

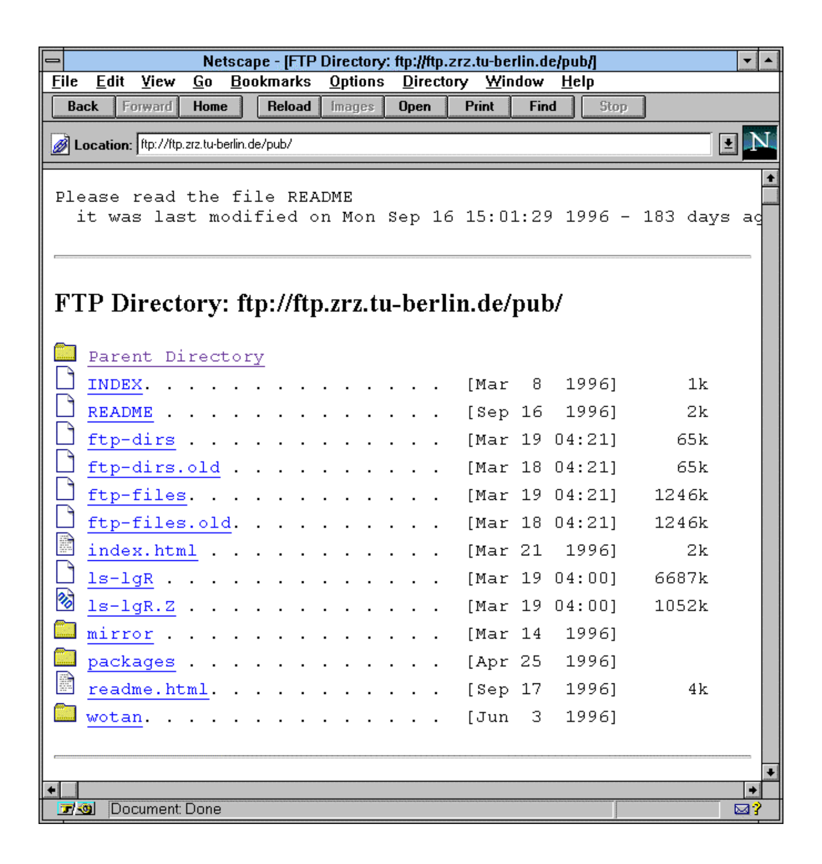


Abbildung 4: Welche Dateinamen befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub" ? (im März 1997)

Bemerkung: In der Ergebnismenge wird eine Unterscheidung zwischen den unterschiedlichen Eintragstypen in Form unterschiedlicher Ikonen vorgenommen; weiterhin werden zusätzliche Angaben dargestellt: letzter Modifikationszeitpunkt, Größe des Eintrag in Kbytes, Verweis auf den gezeigten Eintrag

2. Welche Dateinamen befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub" oder in einem Verzeichnis unterhalb von "pub"? Das Ergebnis soll alphabetisch nach den Namen geordnet werden.
⇒ ftp ftp.zrz.tu-berlin.de
⇒ cd pub
⇒ ls -R
Diese Anfrage ist mit einer WWW-Benutzeroberfläche als URL nicht möglich.

3. Welche Dateien (Eintragstyp, Benutzer- und Gruppenrechte, Zahl der Links, Benutzername, Gruppenname, Größe, letzter Modifikationszeitpunkt, Name) befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub"? Das Ergebnis soll alphabetisch nach den Namen geordnet werden.
⇒ ftp ftp.zrz.tu-berlin.de
⇒ cd pub
⇒ dir
Diese Anfrage ist mit einer WWW- Benutzeroberfläche als URL nicht möglich.

4. Welche Dateien (Art, Benutzer- und Gruppenrechte, Zahl der Links, Benutzername, Gruppenname, Größe, letzter Modifikationszeitpunkt, Name) befinden sich auf dem FTP-Server

"ftp.zrz.tu-berlin.de" im Verzeichnis "pub"? Das Ergebnis soll nach dem letzten Modifikationszeitpunkt geordnet werden.
⇒ ftp ftp.zrz.tu-berlin.de
⇒ cd pub
⇒ ls -lat
Diese Anfrage ist mit einer WWW- Benutzeroberfläche als URL nicht möglich.

5. Welche Zeichenfolge befindet sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub/" in dem Inhalt der Datei mit dem Namen "README" ?
⇒ ftp ftp.zrz.tu-berlin.de
⇒ cd pub/
⇒ get README
⇒ quit
⇒ vi README
oder mit einer WWW-Benutzeroberfläche:

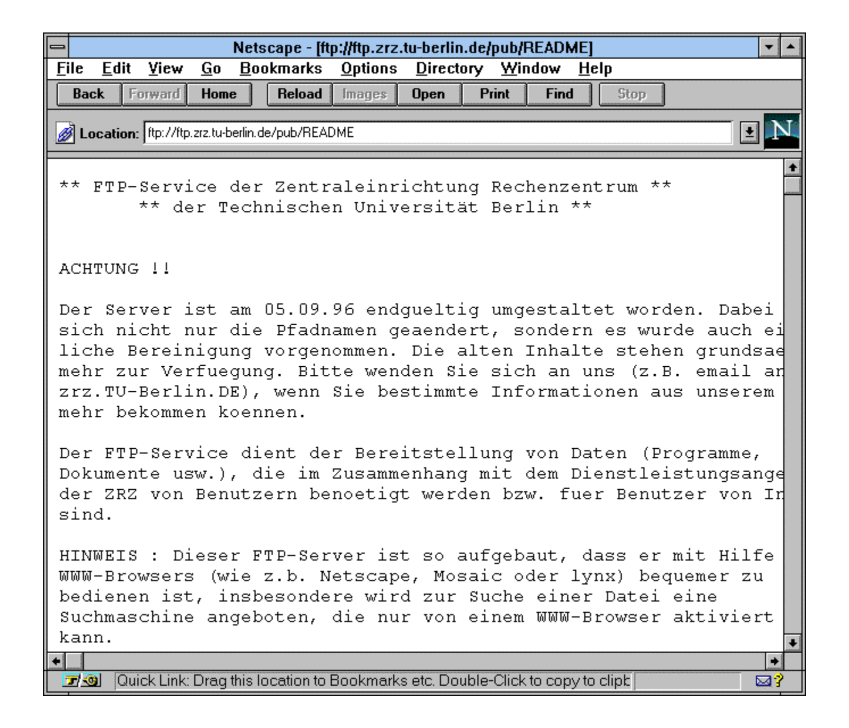


Abbildung 5: Welche Zeichenfolge befindet sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub/" in dem Inhalt der Datei mit dem Namen "README"? (im März 1997)

6. Welche Dateien befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub" oder in einem Verzeichnis unterhalb von "pub", deren Benutzerkennung "ftpadm" ist?
⇒ ftp ftp.zrz.tu-berlin.de
⇒ cd pub
⇒ ls -lR out.txt
⇒ ! grep ftpadm out.txt
Bei dieser Anfrage werden jedoch zusätzlich Einträge als Ergebnis geliefert, die in irgendeinem Feld die Zeichenfolge "ftpadm" enthalten.

7. Welche Dateinamen vom Typ Dateiverzeichnis befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub" oder in einem Unterverzeichnis von "pub" ?

8. Welche Dateien befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub" oder in einem Verzeichnis bis zur Tiefe 3 unterhalb von "pub"? Das Ergebnis soll alphabetisch nach den Namen geordnet werden.

9. Welche Dateien befinden sich auf dem FTP-Server "ftp.zrz.tu-berlin.de" im Verzeichnis "pub" und haben in ihrem Inhalt die Zeichenfolge "bla" enthalten?

Die Anfragen 7, 8 und 9 sind mit einer kommandoorientierten FTP-Anwendung und mit einer WWW-Benutzeroberfläche nicht möglich.

Leistungsgrenzen

- 7-Bit-bzw. 8-Bit-Zeichensatz
- Für den FTP-Zielrechner ist eine Benutzerkennung erforderlich
- Keine regulären Ausdrücke möglich
- Spezifizierung von Attributen ist bei der Suche nur teilweise möglich
- Bestand ist auf ein FTP-System beschränkt
- Keine einheitliche booleschen Verknüpfungen möglich
- Keine Operatoren für Nachfahren, Vorfahren, Verweise etc.

3.1.3.4 Alex

Alex (Cate (1992)) stellt Datenbestände von FTP-Servern zur Nutzung in lokalen Dateisystemen zur Verfügung. Dadurch können Anfragekommandos des lokalen Betriebssystems für die Suche im entfernten FTP-Bestand benutzt werden. Voraussetzung für die Benutzung von Alex ist das verteilte Dateisystem NFS. NFS ist sowohl auf Client- als auch auf Server-Seite in vielen Betriebssystemen verfügbar (Unix, Netware, MS-Windows, VAX/VMS, MVS, DOS etc.). NFS verfügt über einen Cache-Mechanismus, mit dem die Zugriffe auf Einträge in entfernten Dateisystemen effizienter durchgeführt werden können.
Die Bereitstellung von Alex kann beispielsweise auf folgende Weise durchgeführt werden:
Auf der Clientseite wird ein Dateiverzeichnis "/alex" eingerichtet. Mit dem Kommando: "mount -o time=30, retrans=300, soft, intr alex.sp.cs.cmu.edu:/ /alex" wird das Dateisystem des entfernten Alex-Servers in das lokale Dateisystem in das Verzeichnis "/alex" eingehängt. Um einen effizienten Zugriff zu gewährleisten, sollte möglichst ein "in der Nähe befindlicher Alex-Server" eingesetzt werden.
Das über NFS in das lokale System eingebrachte NFS-Dateisystem des Alex-Rechners ist hierarchisch nach Internet-Domänen und den entfernten FTP-Dateiverzeichnissen strukturiert.
Dateiverzeichnisse der obersten Hierarchiestufe haben den Namen der ersten Domainhierarchie (beispielsweise de, edu, usw.), Dateiverzeichnisse der zweiten Hierachiestufe den Namen der zweiten Domainhierarchie (beispielsweise tu-berlin, th-darmstadt, usw.), Dateiverzeichnisse der dritten Hierarchiestufe den Namen der dritten Domainhierarchie (beispielsweise cs, informatik, fb12, usw.) usw. Nach der Hierarchiestufe der Domainrechnernamen (bei Universitäten meistens die Hierarchiestufe 4: ftp, asksam, usw.) folgt in den nächstniedrigeren Hierarchiestufen die Dateiverzeichnisstruktur des jeweiligen FTP-Domainrechners bis hin zu den Dateien.
Ein Beispiel für ein durch Alex zur Verfügung gestelltes Dateiverzeichnis ist: "/alex/de/tu-berlin/zrz/ftp/pub/"

Auf das durch Alex angebundene globale FTP-Dateiverzeichnis kann mit den lokalen Betriebssystemkommandos zugegriffen werden. Anfragekommandos unter Unix sind beispielsweise: "dir", "egrep", "file", "find", "grep", "locate", "ls", "pwd", "stat". Es gilt der 8-Bit-Zeichensatz. Im Vergleich zum lokalen Zugriff muß allerdings eine starke Verminderung der Effizienz in Kauf genommen werden. Das Kommando "find /alex/de -name gnu -print" ausgeführt auf einem Rechner an der TU Berlin zu normalen Betriebszeiten würde ohne Cache ca. 10000 Dateien pro Stunde liefern.
Eine Weiterentwicklung von Alex, Archia (Cate (1992)), bindet Archie an Alex an. Mit Archia ist es möglich, nach Alex-Einträgen zu suchen.
Eine andere Weiterentwicklung sieht vor, mit Hilfe von Volltextdatenbanksystemen den FTP-Bestand auf einem Alex-Rechner zu indexieren.

Leistungsgrenzen

Es gelten dieselben Grenzen wie bei FTP abgesehen von der Beschränkung des Bestands auf ein FTP-System.

3.1.3.5 Archie

Archie ist ein Informationssystem, das Beschreibungen von Datenbeständen öffentlicher FTP-Server zugänglich macht.
Archie baut zu festgelegten Zeiten eine Datenbank der Datei/Verzeichnis-Einträge über ausgewählte FTP-Server (ca. 1500 nach Gilster (1995) S. 17) automatisch auf (mit Hilfe der dort vorliegenden rekursiven "ls -lR"-Auflistungen). Dabei werden Konsistenzprüfungen durchgeführt. In der Datenbank werden zu jedem Eintrag die folgenden Attribute festgehalten:
FTP-Server-Name (DNS-Name), FTP-Server-Nummer (IP-Nummer), Zeitpunkt der Aufnahme in die Datenbank, Name des nächsthöheren Verzeichnisses, Eintragstyp (Datei, Verzeichnis), Benutzer/Gruppenrechte, Größe in Bytes, letzter Modifikationszeitpunkt (Datum, Uhrzeit), Name des Eintrags.
Die Anwendungsprogramme für Archie (Deutsch, Emtage (1992), Bunyip (1996), Liebe (1995)), Xarchie (Ferguson (1994)), Ftp-Search (Bakken (1995)) sind sich semantisch ähnlich.
Die mächtigsten Anfragemöglichkeiten bietet die Telnet-Anwendung Archie. Sie wird anhand von Beispielen vorgeführt:

Beispielanfragen

Anfragen nach Dateien mit Attributen
1. Welche Dateien enthalten im Dateinamen die Zeichenfolge "gnu" ? (Groß/Kleinschreibung wird nicht unterschieden)
⇒ telnet archie.th-darmstadt.de
⇒ set search sub
⇒ find gnu

2. Welche Dateien enthalten im Dateinamen die Zeichenfolge "Gnu"?
(Groß/Kleinschreibung wird nicht unterschieden)
⇒ telnet archie.th-darmstadt.de
⇒ set search subcase
⇒ find gnu

3. Welche Dateien haben exakt den Namen "gcc" ? (Groß/Kleinschreibung wird unterschieden)
⇒ telnet archie.th-darmstadt.de
⇒ set search exact
⇒ find gcc

4. Welche Dateien enthalten im Dateinamen die Zeichenfolge "gnu" (Groß/Kleinschreibung wird nicht unterschieden) und befinden sich auf einem FTP-Server in Deutschland (in der Domain "de") ?
⇒ telnet archie.th-darmstadt.de
⇒ set match_domains de
⇒ find gnu
Eine Menge von Rechnern kann durch das Trennsymbol ":" angegeben werden, beispielsweise durch "set match_domains edu:mil:com:gov:us"

5. Welche Dateien enthalten im Dateinamen die Zeichenfolge "gcc" (Groß/Kleinschreibung wird nicht unterschieden) und befinden sich in einem Verzeichnis, dessen Name die Zeichenfolge "gnu" (Groß/Kleinschreibung wird nicht unterschieden) enthält ?
⇒ telnet archie.th-darmstadt.de
⇒ set match_path gnu
⇒ find gcc

Anfragen nach Dateien mit eingeschränkten regulären Ausdrücken
6. Welche Dateien haben einen Dateinamen, der mit der Zeichenfolge "Gnu" beginnt ? (Groß/Kleinschreibung wird unterschieden)
⇒ telnet archie.th-darmstadt.de
⇒ set search regex
⇒ find ^Gnu

7. Welche Dateien enthalten im Dateinamen den regulären Ausdruck
"[Ff]ile ?[Ss]ystem?" (Dateinamen, die mit dem Zeichen "F" oder "f" beginnen, gefolgt von der Zeichenfolge "ile", gefolgt von 0 bis n beliebigen Zeichen, gefolgt von den Zeichen "S" oder "s", gefolgt von der Zeichenfolge "ystem", gefolgt von 0 bis n beliebigen Zeichen) ?
(Groß/Kleinschreibung wird unterschieden)
⇒ telnet archie.th-darmstadt.de
⇒ set search regex
⇒ find .*[Ff]ile.*[Ss]ystem.*

Ordnung und Beschränkung von Suchergebnissen
8. Welche Dateien enthalten im Dateinamen die Zeichenfolge "gnu" (Groß/Kleinschreibung wird nicht unterschieden) ? Das Ergebnis wird alphabetisch nach dem Dateinamen geordnet.
⇒ telnet archie.th-darmstadt.de
⇒ set sortby filename
⇒ find gnu
Weitere Ordnungskriterien sind beispielsweise:
a) nach Rechnernamen ("hostname")
b) nach Größe des Eintrags ("size")
c) nach dem letzten Modifikationszeitpunkt ("time")

d) ungeordnet ("none")

9. Welche Dateien enthalten im Dateinamen die Zeichenfolge "gnu" (Groß/Kleinschreibung wird nicht unterschieden) ? Die Größe der Ergebnismenge wird auf "25" begrenzt.
⇒ telnet archie.th-darmstadt.de
⇒ set maxhits 25
⇒ find gnu
Weitere Mengenbeschränkungen sind beispielsweise:
a) maximale Zahl von Einträgen gleichen Namens ("maxhitspm")
b) maximale Zahl von Einträgen unterschiedlichen Namens ("maxmatch")

Leistungsgrenzen

- 8-Bit-Zeichensatz
- Spezifizierung von Attributen nur teilweise möglich
- Reguläre Ausdrücke (eingeschränkt) können nur im Attribut "Dateiname" verwendet werden
- Keine einheitliche boolesche Verknüpfung möglich
- Keine Operatoren für Nachfahren, Vorfahren, Verweise etc.

3.1.4 E-Mail-Suche

Electronic Mail (E-Mail) ist ein Informations- und Kommunikationssystem, das den Austausch von Nachrichten (E-Mails) über Rechnernetzwerke ermöglicht. Im Internet wird E-Mail überwiegend über das "Simple Mail Transfer Protocol (SMTP)" nach RFC 822 benutzt. SMTP ist momentan auf den 7-Bit Zeichsatz beschränkt. Weiterhin wird nicht spezifiziert, wie Nicht-Text-Datentypen (Ton, Graphik, Video etc.) übertragen werden können. Eine Lösung für beide Probleme wird mit MIME (RFC 1521, RFC 1522) geliefert. Mit MIME wrd eine E-Mail in mehrere Felder aufgetrennt:
1. SMTP-Felder: Action, Copy, Date, Expires, From, In-reply-to, Message-Id, Precede(n)ce, Priority, Received, Reply-to, Subject, To
2. Weitere Felder: Mime-Version, Content-Type, Content-Transfer-Encoding, Content-Id, Content-Description
3. Eigendefinierte Felder
Für SMTP und MIME existieren eine Vielzahl von Anwendungsprogrammen.
Ein weiteres System ist das "Message Handling System (MHS)" nach X.400.
E-Mail-Anwendungsprogramme bieten die Möglichkeit, Mengen von E-Mails (sogenannte Folder) zu benennen. Für die empfangenen und versendeten Nachrichten existieren in den Anwendungsprogrammen Standard-Folder (beispielsweise received und sent). Subfolder können rekursiv bestimmt werden. Eine Suche nach E-Mails kann mit Attributen durchgeführt werden.
E-Mails können auch mit Volltextsuchsystemen indexiert werden. Diese Systeme bieten dann Anfragemöglichkeiten dieser Systeme (s.u.).

Beispielanfragen

1. Welche E-Mails liegen im Folder "received" ?
⇒ elm

2. Welche E-Mails liegen im Folder "sent" ?

⇒ elm
⇒ c
⇒ =sent

3. Welche E-Mails im Folder "privat" wurden von der Person mit dem Nachnamen "Mueller" an mich gesendet?
⇒ elm
⇒ c
⇒ =privat
⇒ l
⇒ from Mueller

4. Welche E-Mails im Folder "privat" enthalten im Attributwert des Attributs "Subject" die Zeichenfolge "www"
⇒ elm
⇒ c
⇒ =privat
⇒ l
⇒ subject www

3.1.5 Artikelsuche

News-Systeme wie zum Beispiel Netnews (Storm (1995)), Tin (Lea (1995)) oder Dejanews (2000a) bieten Anfragemöglichkeiten nach Newsgruppen und Artikeln (News).
Dejanews bietet die mächtigsten Anfragemöglichkeiten.

Beispielanfragen

Anfragen nach Newsgruppen
1. Welche Newsgruppen existieren?
Netnews:
⇒ nn
⇒ Y
News-URL:
⇒ news:* (Ausgabe als Baum)
Dejanews-Anfrage:
⇒ http://www.dejanews.com/toplevel.html (Ausgabe der obersten Hierarchiestufe)

2. Welche Newsgruppe ist alphabetischer Nachfolger der selektierten Newsgruppe?
Netnews:
⇒ nn
⇒ N (Newsgruppe wird ausgewählt bzw. aktiviert) oder A (Newsgruppe wird nicht ausgewählt bzw. aktiviert)

Tin:
⇒ tin
⇒ j (Newsgruppe wird nicht ausgewählt bzw. aktiviert)

3. Welche Newsgruppe ist der alphabetische Vorgänger der selektierten Newsgruppe ?
Netnews:
⇒ nn
⇒ P (Newsgruppe wird ausgewählt bzw. aktiviert), B (Newsgruppe wird nicht ausgewählt bzw. aktiviert)
Tin:
⇒ tin
⇒ k (Newsgruppe wird nicht ausgewählt bzw. aktiviert)

4. Welche Newsgruppen sind die Nachfolger der selektierten Newsgruppen (so viele, wie auf den Bildschirm darstellbar) ?
Tin:
⇒ tin
⇒ Ctrl-d

5. Welche Newsgruppen sind die Vorgänger der selektierten Newsgruppen (so viele, wie auf den Bildschirm darstellbar) ?
Tin:
⇒ nn
⇒ Ctrl-u

6. Welche Newsgruppen enthalten Artikel, die die Zeichenfolge "www" im Attributwert des Attributs "Inhalt" enthalten (geordnet nach der Häufigkeit des Vorkommens) ?
Dejanews-Anfrage:
⇒ http://search.dejanews.com/query_profile.xp?query=www

Anfragen nach Newsgruppen mit Volltext
Bei Netnews wird das Ergebnis alphabetisch sortiert. Bei Eingabe von "n" wird die alphabetisch nachfolgende Newsgruppe angezeigt.
7. Welche Newsgruppen enthalten im Attribut "Name" die Zeichenfolge "bln.net.www"?
Netnews:
⇒ nn
⇒ G bln.net.www
Dejanews-Anfrage mit einer WWW-Benutzeroberfläche:
⇒ http://search.dejanews.com/bg.xp?level=bln.net.www

8. Welche Newsgruppen enthalten im Attribut "Name" die Zeichenfolge "bln" am Beginn?
Netnews:
⇒ nn
⇒ G bln*
oder
⇒ nngrep bln
News-URL:
⇒ news:bln*
Dejanews-Anfrage:
⇒ http://search.dejanews.com/bg.xp?level=bln

9. Welche Newsgruppen enthalten im Attribut "Name" die Zeichenfolge "bln" am Ende ?
Netnews:
⇒ nn
⇒ G bln$
Nngrep:
⇒ nngrep bln$
Dejanews-Anfrage: nicht möglich

Anfragen nach Artikeln (News)
10. Welche Artikel existieren (alle Artikel aller Newsgruppen) ?
Netnews:
⇒ nn -Xm *

11. Welche Artikel enthält die Newsgruppe "bln.net.www" ?
Netnews:
⇒ nn
⇒ G bln.net.www
Tin:
⇒ tin
⇒ g bln.net.www Space
News-URL:
⇒ news:bln.net.www

12. Welcher ist der gerade selektierte Artikel ?
Netnews:
⇒ nn
⇒ .
Wenn die ausgewählten Artikel sich zeitlich am Ende/am Anfang einer Newsgruppe befinden, werden die Artikel der alphabetisch nachfolgenden/vorangehenden Newsgruppe als Antwort gegeben. Diese Anfragen werden auch lineares Blättern genannt.

13. Welcher Artikel ist der Nachfolger des selektierten Artikels ?
Netnews:
⇒ nn
⇒ ,
Tin:
⇒ tin
⇒ j

14. Welcher Artikel ist der Vorgänger des selektierten Artikels ?
Netnews:
⇒ nn
⇒ /
Tin:
⇒ tin
⇒ k

15. Welche Artikel sind Nachfolger der selektierten Artikel in der selektierten Newsgruppe (so viele, wie auf den Bildschirm darstellbar) ?
Netnews:
⇒ nn
⇒ Space oder >
Tin:
⇒ tin
⇒ Ctrl-d

16. Welche Artikel sind die Vorgänger der selektierten Artikel in der selektierten Newsgruppe (so viele, wie auf den Bildschirm darstellbar) ?
Netnews:
⇒ nn
⇒ <
Tin:
⇒ tin
⇒ Ctrl-u

17. Welche Artikel sind die zeitlich letzten in der selektierten Newsgruppe:
Netnews:
⇒ nn
⇒ $

18. Welche Artikel haben den Namen "czyborra960131090210354@titanic.cs.tu-berlin.de" ?
News-URL:
⇒ news:czyborra960131090210354@titanic.cs.tu-berlin.de (News Nr. 10354 vom 31.1.96 um 9.02 Uhr auf titanic.cs.tu-berlin.de von der Benutzerkennung czyborra abgeschickt)

19. Welcher Artikel ist der zeitlich erste in der selektierten Newsgruppe ?
Netnews:
⇒ nn
⇒ a
Tin:
⇒ tin
⇒ 1

20. Welche Artikel sind zeitlich die ersten 4 in der selektierten Newsgruppe?
Netnews:
⇒ nn
⇒ a-d
Tin: nicht möglich

Anfragen nach Artikeln mit Attributen

21. Welche Artikel haben denselben Wert für das Attribut "Titel" wie der Artikel "a" ?
Netnews:
⇒ nn
⇒ a*

Tin:
⇒ tin
⇒ l (Liste der Threads)

22. Welche Artikel enthalten im Attribut "Titel" die Zeichenfolge "www"?
Netnews:
⇒ nn
⇒ = www
Tin:
⇒ tin
⇒ / www

23. Welche Artikel enthalten im Attribut "Titel" die Zeichenfolge "in", gefolgt von dem Zeichen "t" oder "d"?
Netnews:
⇒ nn
⇒ = in[dt]
Tin: nicht möglich

24. Welche Artikel enthält die selektierte Newsgruppe ?
Netnews:
⇒ nn
⇒ = .
Tin:
⇒ tin
⇒ CR

25. Welche Artikel der Newsgruppe "bln.net.www" enthalten im Attribut "Autor" die Zeichenfolge "josef" (Groß/Kleinschreibung wird nicht unterschieden)?
Netnews:
⇒ nn -Xxm -njosef bln.net.www

26. Welche Artikel der Newsgruppe "bln.net.www" enthalten im Attribut "Subject" die Zeichenfolge "provider" ?
Netnews:
⇒ nn -Xxm -sprovider bln.net.www

27. Welche Artikel der Newsgruppe, deren Name die Zeichenfolge "bln" enthält, enthalten im Attribut "Titel" die Zeichenfolge "tu"?
Netnews:
⇒ nn
⇒ G bln
⇒ y(es) zur Auswahl der gewünschten Newsgruppe
⇒ s tu
oder
⇒ nn -stu bln (keine regulären Ausdrücke möglich)
Tin: nicht möglich

28. Welche Artikel erschienen am 10.2.1996 in der Newsgruppe "bln.net.www" ?
Dejanews-Anfrage:
⇒ http://search.dejanews.com/filter.xp?
groups=bln.net.www
&dates=1996%2F02%2F10
&fromdate=&todate=
&authors=&subjects=

29. Welche Artikel in der Newsgruppe "bln.net.www" enthalten im Attribut "Titel" oder "Inhalt" die Zeichenfolge "www" ?
Dejanews-Anfrage mit einer WWW-Benutzeroberfläche:
⇒ http://search.dejanews.com/filter.xp?
groups=bln.net.www
&dates=&fromdate=
&todate=&authors=
&subjects=www
⇒ in der Suchmaske die Zeichenfolge "www" eingeben
Bemerkung: Das Ergebnis ist nicht korrekt. Die Ergebnismenge enthält andere Artikel als in der Anfrage gewünscht.

30. Welche Artikel haben das Erstellungsdatum "18.09.1996" und enthalten im Attribut "Titel" die Zeichenfolge "www"?
Dejanews-Anfrage mit einer WWW-Benutzeroberfläche:
⇒ http://search.dejanews.com/filter.xp?
groups=
&dates=1996%2F09%2F18
&fromdate=&todate=&authors=&subjects=www

Leistungsgrenzen

Beispielanfragen, die in den News-Systemen nicht gestellt werden können:
1. Welche Artikel der Newsgruppen, die in dem Attribut "Newsgruppenname" die Zeichenfolge "bln" enthalten, enthalten im Attribut "Titel" die Zeichenfolge "WWW" (Groß/Kleinschreibung wird unterschieden) und sind vom Autor "josefw@cs.tu-berlin.de" oder "czyborra@cs.tu-berlin.de" oder "bene@cs.tu-berlin.de" verfasst ?
2. Welche Dokumente verweisen auf den Artikel "czyborra###374654657@titanic.cs.tu-berlin.de" ?
3. Welche Dokumente verweisen auf Artikel vom Autor "josefw@cs.tu-berlin.de" am "10.2.1996" ?
4. Welche Artikel verweisen auf den selektierten Artikel und verweisen wiederum auf diese Artikel etc. bis zur Stufe "4" ?

3.1.6 Menüsuche

Gopher wurde Anfang 1991 an der Universität von Minnesota (USA) entwickelt (Boden et.al. (1994)). Wörtlich bedeutet Gopher soviel wie Beutelratte. Eine Beutelratte ist auch das Maskottchen der Universität von Minnesota. Weiterhin bedeutet Gopher in einer weiteren Übersetzung rastlos umherirrender Bürobote. Die Aussprache von Gopher paßt auch auf "go for (something)".

Grundlage aller Gophersysteme ist das Internet Gopher Protocol (RFC 1436), mit dem Information hierarchisch über sogenannte Menüs angeboten wird. Mit den einzelnen Menüpunkten kann auf verschiedene Gopher-Datentypen wie z.B. Text, Bild, Verzeichnis, Fehlermeldung, binäre Datei etc., die weltweit verteilt auf Rechnern vorliegen, zugegriffen werden. Mit den Menüpunkten kann auf Informationssysteme wie z.B. Indexsuche, Telefonbuchsuche, Telnet, Ftp, Archie, oder Wais zugegriffen werden.
Seit 1993 wird das Gopher+-Protokoll verwendet, das zu den einzelnen Menüpunkten weitere Felder zur Verfügung stellt wie z.B. Größe, Autor, Zusammenfassung, oder Menüpunkte für andere Sprachen. Neue Feldtypen sind Bilder in Bitmap-Format, Bewegtbilder und Audiodateien.
Das Gopher-Protokol ist inzwischen durch den Datentyp HTML erweitert worden.
Mit den Informationssystemen Veronica (Foster (1994a), Foster (1994b)) und Jughead (Jones (1994)) kann mit Stichwörtern nach Gopher-Menüpunkten gesucht werden. Veronica und Jughead sind Subsysteme von Gopher und können deshalb nur mit einer Gopher-Anwendung benutzt werden. Im Mittelpunkt von Veronica und Jughead steht eine zentrale Datenbank, die an der Universität von Nevada verwaltet wird und mittels FTP ca. alle 1-2 Wochen aktualisiert wird. Im Juni 1994 enthielt die Datenbank ca. 10 Millionen Gopher-Einträge verschiedenen Gophertyps von 5500 Gopher-Servern. Kopien der Datenbank existieren u.a. in Köln.
Benutzeroberflächen für Gopher sind beispielsweise: gopher, xgopher (Tuchman (1993)) und WWW-Benutzeroberflächen.

Beispielanfragen

Anfragen nach Gopher-Menüs mit Attributen
1. Welche Gopher-Menüs enthalten im Attribut "URL" die Zeichenfolge "gopher://gopher.cs.tu-berlin.de" ?
Gopher-URL:
⇒ gopher://gopher.cs.tu-berlin.de

2. Welche Gopher-Menüs enthalten im Attribut "Titel" exakt die Zeichenfolge "veronica"?
Veronica:
⇒ veronica
⇒ veronica
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?veronica

3. Welche Gopher-Menüs enthalten im Attribut "Titel" eine Zeichenfolge, die mit "fahr" beginnt?
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?fahr*

4. Welche Gopher-Menüs enthalten im Attribut "Titel" die Zeichenfolge "veronica" und im Attribut " Typ" die Zeichenfolge " Verzeichnis " ?
Veronica:
⇒ veronica
⇒ veronica -t1
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?veronica+-t1

Boolesche Anfragen
5. Welche Gopher-Menüs enthalten im Attribut "Titel" die Zeichenfolge "veronica" oder die Zeichenfolge "jughead" exakt ?
Veronica:
⇒ veronica
⇒ veronica or jughead
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?veronica+or+jughead

6. Welche Gopher-Menüs enthalten im Attribut "Titel" die Zeichenfolge "veronica" und die Zeichenfolge "jughead" exakt ?
Veronica:
⇒ veronica
⇒ veronica and jughead
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?veronica+and+jughead

7. Welche Gopher-Menüs enthalten im Attribut "Titel" die Zeichenfolge "veronica" aber nicht die Zeichenfolge "mueller" ?
Veronica:
⇒ veronica
⇒ veronica not mueller
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?veronica+not+mueller

Komplexe Anfragen
8. Welche Gopher-Menüs enthalten im Attribut "Titel" Zeichenfolgen beginnend mit "vero" oder "jug" und im Attribut "Typ" die Zeichenfolge " Verzeichnis " ?
Veronica:
⇒ veronica
⇒ (vero* or jug*) -t1
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?%28vero+or+jug%29+-t1

9. Welche Gopher-Menüs vom Typ "Verzeichnis" oder "Textdatei" oder "Sound" enthalten im Attribut "Titel" Zeichenfolgen beginnend mit "vero" oder "jug" (die Größe der Ergebnismenge soll maximal "30" sein) ?
Veronica:
⇒ veronica
⇒ (vero* or jug*) -ts01 -m30
Veronica-URL:
⇒ gopher://veronica.uni-koeln.de:2347/7?%28vero+or+jug%29+-ts01+-m30

Leistungsgrenzen
Folgende Anfrage kann mit Gopher nicht gestellt werden:
1. Welche Gophermenüs enthalten in dem Attributwert des Attributs "Titel" die Zeichenfolge "ö" oder "und" ?

3.1.7 Suche nach Dokumenthinweisen

Fast alle großen wissenschaftlichen Bibliotheken bieten ihre Bestände mittlerweile im Internet an. In letzter Zeit ist eine Zentralisierung zu beobachten. Informationsbestände wachsen zu Informationsverbünden zusammen, einzelne Bibliothekssysteme werden zu Großbibliotheks-systemen zusammengeführt. Möglich ist dies durch die verteilte Datenverarbeitung im Inter-net und durch die Entwicklung von internationalen Bibliotheksstandards wie z.B. Z39.50 (Z39.50-1995). Neuerdings bieten Bibliotheken und Bibliotheksverbünde Bibliothekskatalo-ge (OPAC's) im WWW an. Dies sind z.B. Bibliotheken einzelner Länder (Library of Con-gress, Deutsche Bibliothek etc.), regionale Verbünde, einzelne wissenschaftliche Bibliotheken etc. Auflistungen von Bibliotheken (Deutschland, international) finden sich unter HBZ (2001a) und HBZ (1997b). OCLC (2001) hat es sich zum Ziel gesetzt, alle Bibliothekskatalo-ge weltweit zu vereinen und bietet einen OPAC mit 33 Millionen Einträgen.
Buchhandlungen und Verlage bieten Suchmöglichkeiten für die Verzeichnisse lieferbarer Bü-cher an. Eine umfangreiche Auflistung findet sich in HBZ (1997d) und HBZ (1997e).
In den Bibliothekssystemen werden Bibliothekskataloge mit Dokumenthinweisen gespeichert. Volltextangebote entstehen bisher nur in kleinerem Rahmen (Dissertationen, Zeitschriften etc.).
Für die Dokumenthinweise existieren unterschiedliche Felder (vgl. ISO 8777, UNIMARC (1997), USMARC, ISO 2709, MAB (1997), Daniel et.al. (1995) und DublinCore (2001)). Für den Buchhandel gelten zusätzliche Attribute: Preis, Anzahl der Seiten, Lieferzeitpunkt etc.

3.1.8 Volltextsuche

3.1.8.1 WAIS

Grundlage von WAIS (Wide Area Information Server) ist das Protokoll Z39.50 (Z39.50-1995). Z39.50 spezifiziert Prozeduren und Formate für den Austausch von Nachrichten zwi-schen Z39.50-Client und -Server. Der Client übermittelt Anfragen an den Server. Dieser bear-beitet die Anfrage und übermittelt das Ergebnis an den Client zurück. Der Client selektiert dann Elemente der Ergebnismenge.
Z39.50 hat von 1981 bis heute drei Stufen durchlaufen: Version 1 (1988), Version 2 (1992) und Version 3 (1995).
Version 3 bietet folgende Merkmale:

- Unterstützung von Sitzungen als zeitliche Folge von Operationen. Relevanzfeedback ist dadurch möglich. Operationen können gleichzeitig durchgeführt werden.
- Indexierung geschachtelter Dokumente (SGML etc.). Felder werden automatisch erkannt.
- Formate für die Ergebnismengen (z.B. USMARC, UKMARC etc.)
- Benennung von Ergebnismengen
- Auftrennung von Ergebnismengen in Teilmengen (bei ineffizienten Datenverbindungen)
- Client-Authentifizierung durch den Server
- Attribute können Datentypen zugeordnet werden (Integer, Date, Char, etc.)
- Unterschiedliche Attributmengen (z.B. Bib-1 oder STAS (STAS (1996))
- Unterschiedliche boolesche Anfrageformate: umgekehrt polnische Notation ("information retrieval and"), ISO 8777 etc.
- Verwendung von Attributen unterschiedlicher Attributmengen in einer Anfrage
- Anfragen an Ergebnismengen
- Ordnung von Ergebnismengen

- Anfragen nach Attributen von Z39.50 Servern (verwendete Datenbasen, Attributmengen, Server-Administrator etc.)
- Zeichensatzangaben für Z39.50-Sitzungen
- Unterstützung von ISO 10162/10163

WAIS ist ein Z39.50-basiertes Informationssystem zur Volltextsuche und zur Suche mit Attributen in verteilten WAIS-Beständen. WAIS ist eine Teilmenge von Z39.50 Version 1, erweitert dieses jedoch durch Sitzungen.
WAIS ist in einer Client-Server-Architektur mit dem WAIS-Protokoll realisiert. Seit 1992 wird die nicht kommerzielle Weiterentwicklung von WAIS unter dem Namen FreeWAIS (FreeWAIS (2001)) fortgeführt.

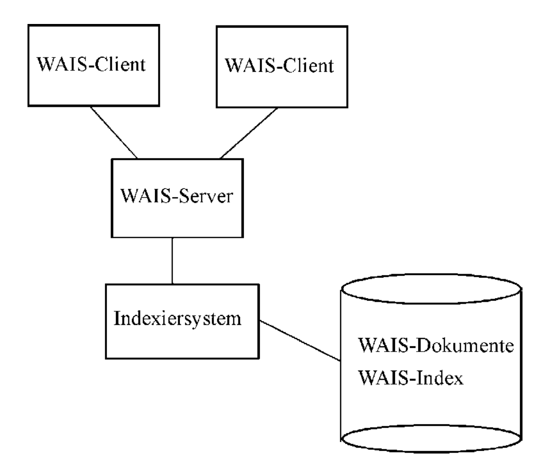


Abbildung 6: Architektur von WAIS

Der Dokumentenbestand zusammen mit seinem Index wird Datenbasis genannt. Dokumente können unterschiedlichen Typen zugeordnet werden. WAIS baut mit Hilfe des Indexiersystems einen Index über den Bestand auf.
Weltweit existiert eine Vielzahl von WAIS-Datenbasen. Für eine globale WAIS-Suche wird zunächst eine Auswahl von WAIS-Datenbasen durchgeführt. Dafür existieren WAIS-Datenbasen von WAIS-Datenbasen (directory-of-servers). Beim Einrichten von WAIS-Datenbasen wird eine Beschreibung in Form von Feldern angelegt: Rechnername, IP-Adresse, Serverport, Name der Datenbasis, E-mail-Adresse des Systemverwalters, inhaltliche Attribute, inhaltliche Kurzbeschreibung. Mit Attributen kann nach solchen WAIS-Datenbasen gesucht werden.
Momentan kann die Anfrage nach WAIS-Datenbanken weltweit nur auf einem inaktuellen Bestand durchgeführt werden, da die vormals mit der zentralen Sammlung von WAIS-Datenbasen beschäftigten Organisationen (WAIS Inc, Thinking Machines Corp., NASA, CNIDR (1996)) diesen Service nicht mehr öffentlich anbieten. Ältere Datenbasen liegen z.B. noch bei Gövert (1996b) und Ardö, Koch (1994) vor. Gövert (1996b) bietet eine einfache Suche mit Suchbegriffen nach WAIS-Datenbasen an. Ergebnis einer Anfrage ist ein Bestand von WAIS-Datenbasen. Dieser kann durch Selektion einzelner Datenbasen weiter eingeschränkt werden. Mit der Anfragesprache von SFgate wird über eine WWW-Benutzeroberfläche in diesem Bestand weiter recherchiert. Bei Ardö, Koch (1994) wird eine automatische Klassifikation von WAIS-Datenbasen durchgeführt. Aus den Beschreibungen der WAIS-Datenbasen werden Wörter aus dem Beschreibungstext, der Liste der Schlüsselworte und aus dem Sachgebietsfeld extrahiert und automatisch in den Sachgebietsbaum eingefügt. Mit Hilfe dieses

Baumes wird über eine WWW-Schnittstelle eine Auswahl von WAIS-Datenbasen vorgenommen. Dieser Bestand kann durch Selektion einzelner Datenbasen eingeschränkt werden. Wie bei Gövert (1996b) kann dann mit der Anfragesprache von SFgate in diesem Bestand recherchiert werden.
FreeWAIS-sf (Fuhr et.al.(1995)) unterstützt wie WAIS das Protokoll Z39.50 Version 1. Im Gegensatz zum WAIS-Protokoll läßt FreeWAIS-sf keine Sitzungen zu.
Mit FreeWAIS-sf kann eine automatische Indexierung mit vordefinierten und eigenerstellten Dokumenttypen durchgeführt werden. Es können beispielsweise multimediale Typen nach MIME-Standard: TEXT, HTML, URL etc. verwendet werden. Weiterhin kann der Indexierer Dokumenttypen mit Hilfe von regulären Ausdrücken definieren.
FreeWais-sf verwendet den 8-Bit-Zeichensatz nach ISO 8859-X.
Den Dokumenten können Attribute unterschiedlichen Typs zugeordnet werden. Für die Attribute existieren Vergleichsprädikate (enthaltensein, gleich, kleiner, größer, phonetische Ähnlichkeit etc), die in den Anfragen benutzt werden können.
Bei der Indexierung wird eine invertierte Liste aller Terme der Datenbasis aufgebaut. Dort wird zu jedem Term eine Menge von Paaren (Dokumentkennung, Gewicht des Terms für dieses Dokument) festgehalten.
Die Gewichtung eines Terms k für das Dokument i (w_{ik}) wird folgendermaßen berechnet:

Dokumente (Anzahl der Dokumente in der Datenbasis)	n
Dokumenthäufigkeit (Anzahl der Dokumente, in denen $term_k$ vorkommt)	$docfreq_k$
inverse Dokumenthäufigkeit	$idf_k = \log_{10}(\frac{n}{docfreq_k})$
Term-Dokumenthäufigkeit (Anzahl des Vorkommens von $term_k$ in $Dokument_i$)	$freq_{ik}$
Maximale Termhäufigkeit (maximale Häufigkeit eines Terms in $Dokument_i$)	$maxfreq_i$
relative Term- Dokumenthäufigkeit	$relfreq_{ik} = \frac{\frac{1}{2} * freq_{ik}}{1 + maxfreq_i}$
inverse Term-Dokumenthäufigkeit	$invfreq_{ik} = relfreq_{ik} * idf_k$
normalisierte Term-Dokumenthäufigkeit	$w_{ik} = \frac{invfreq_{ik}}{\sum_{j=1}^{k} (invfreq_{ik})^2}$

Die Syntax der Anfragesprache von FreeWAIS-sf wird in Gövert, Pfeifer (1996) definiert. Ergebnis einer Anfrage mit freeWAIS-sf ist eine nach Ähnlichkeit zwischen Anfrage und Dokumenten geordnete Menge von Dokumenten. FreeWais-sf verwendet als Ähnlichkeitsfunktion das Produkt von Anfrageterm- und Dokumenttermgewichtung (Pfeifer (1995) S. 28-29). Als boolesche Operatoren werden or (oder), and (und) und not (binäres nicht) unterschieden. Die Berechnung der Ähnlichkeit einer Anfrage q zu einem Dokument d wird folgendermaßen definiert:
Ähnlich (qw_k , d) = $qterm_k * w_{ik}$ für i=1..n
Ähnlich (q_1 or q_2 , d) = Ähnlich (q_1 , d) + Ähnlich (q_2 , d)
Ähnlich (q_1 and q_2 , d) = min (Ähnlich (q_1 , d) , Ähnlich (q_2 , d))
Ähnlich (q_1 not q_2 , d) = min (Ähnlich (q_1 , d) , (1 - Ähnlich (q_2 , d)))

Mit einem Client wie z.B. waissearch (Waissearch (1992)), waisq (Waisq (1992)), xwais (Xwais (1992)), SFgate (Gövert, Pfeifer (1996), Gövert (1996a)), WAIS-URL (RFC 1738, RFC 1808) etc., wird über das WAIS-Protokoll eine Anfrage an den Anfragedienst gestellt. Dieser liefert über den Index eine geordnete Menge von Dokumentbeschreibungen an den Client zurück.
SFgate ist eine Schnittstelle zwischen dem WWW und FreeWAIS-sf. SFgate stellt logische Datenunabhängigkeit her, indem die in den einzelnen FreeWAIS-sf-Datenbasen benutzten Attribute auf universell einheitliche Attribute (STAS (1996)) abgebildet werden. In STAS (1996) werden ca. 1000 Attribute über eindeutige Nummern und einen Namen (2-5 Großbuchstaben) identifiziert. Dadurch wird dem Benutzer eine von den verschiedenen Datenbanken unabhängige Ebene präsentiert. Anfragen mit Attributen können so einheitlich über mehrere Datenbanken gestellt werden. STAS (1996) unterscheidet beispielsweise die Attribute (Name, Abkürzung, Nummer): Abstract: AB, 62; Application: APP, 3180; Broader Term: BT, 3026; Duration: DUR, 2845; etc.

Beispielanfragen

Im Folgenden werden zur Anfrage in WAIS-Datenbasen die Clients waissearch (freeWAIS (2001)) und SFgate (Fuhr et. al. (1995)) benutzt. Waissearch wird als Unix-Programm gestartet. SFgate wird mit einer URL aufgerufen (hier: http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate. Für Metazeichen wird als Fluchtzeichen das Zeichen "%" gefolgt von dem hexadezimalen ASCII-Zeichencode benutzt (das Zeichen "/" wird z.B. durch die Zeichenfolge "%2F" ausgedrückt).

Anfragen nach WAIS-Datenbasen

Die folgenden beiden Anfrage können gestellt werden, wenn die Datenbasis "directory-of-servers" als Index erstellt wurde (z.B. mit freeWAIS-sf).

1. Welche WAIS-Datenbasen enthalten in einem Attributwert die Zeichenfolge "computer"?
Waissearch:
⇒ waissearch -h ls6.informatik.uni-dortmund.de -p 210 -d directory-of-servers computer
WAIS-URL:
⇒ wais://ls6.informatik.uni-dortmund.de/directory-of-server?computer
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fdirectory-of-servers
&computer

2. Welche WAIS-Datenbasen enthalten im Attribut "Titel" die Zeichenfolge "information"
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fdirectory-of-servers
&title%3Dinformation

Anfragen nach Dokumenten

3. Welche Dokumente der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolge "information" ?
Waissearch:
⇒ waissearch -h ls6.informatik.uni-dortmund.de -p 210 -d bibdb-html information
WAIS-URL:

⇒ wais://ls6.informatik.uni-dortmund.de/bibdb-html?information
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&information

4. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolge "information" (die Größe der Ergebnismenge soll maximal "20" sein; zu jedem Dokument sollen folgende Felder angezeigt werden: Name der Datenbank, Größe in Bytes, Typ und Gewichtung; jedes Element der Ergebnismenge kann durch einen Auswahlknopf für eine spätere Anzeige selektiert werden; die Ausgabe des Ergebnisses soll in englischer Sprache geschehen).
SFgate-Anfrage:
⇒http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&information
&maxhits=20
&range=1
&verbose=1
&multiple=1
&language=english

5. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolge "information retrieval" ?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&%22information%20retrieval%22

6. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolge "information" oder eine zu "information" phonetisch ähnliche Zeichenfolge ?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&%28PHONIX+information%29+%28SOUNDEX+information%29

7. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" haben zwischen den Zeichenfolgen "information" und "retrieval" einen Wortabstand von höchstens 10 Worten ?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&information w/10 retrieval

8. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" haben nach der Zeichenfolge "information" höchstens einen Wortabstand von 10 Worten zur Zeichenfolge "retrieval" ?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&information pre/10 retrieval

9. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolge "information" "20" mal ?
Waissearch und WAIS-URL: nicht möglich
als SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&atleast 20 information

Boolesche Anfragen nach Dokumenten
10. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolgen "information" oder "retrieval" ?
Waissearch :
⇒ waissearch -h ls6.informatik.uni-dortmund.de -p 210 -d bibdb-html information retrieval
oder
⇒ waissearch -h ls6.informatik.uni-dortmund.de -p 210 -d bibdb-html information or retrieval
oder
⇒ waissearch -h ls6.informatik.uni-dortmund.de -p 210 -d bibdb-html "information retrieval"
oder
⇒ waissearch -h ls6.informatik.uni-dortmund.de -p 210 -d bibdb-html "information or retrieval"
WAIS-URL:
⇒ wais://ls6.informatik.uni-dortmund.de/bibdb-html?information+retrieval
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&information+retrieval

11. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolgen "information" und "retrieval" ?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&information+and+retrieval

12. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" enthalten die Zeichenfolge "information" aber nicht die Zeichenfolge "retrieval" ?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&information+not+retrieval

Anfragen in mehreren Datenbasen
13. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" oder in der WAIS-Datenbasis "journals" auf dem WAIS-Server "bib.informatik.uni-dortmund.de" enthalten Zeichenfolgen, die mit "inform" beginnen ?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
?database=bib.informatik.uni-dortmund.de%2Fjournals
&inform*

14. Welche Dokumente in der WAIS-Datenbasis "bibdb-html" auf dem WAIS-Server "ls6.informatik.uni-dortmund.de" oder in der WAIS-Datenbasis "journals" auf dem WAIS-Server "bib.informatik.uni-dortmund.de" enthalten die Zeichenfolgen "information" oder "retrieval" und haben ein Erscheinungsjahr, das größer als "1995" ist?
Waissearch und WAIS-URL: nicht möglich
SFgate-Anfrage:
⇒ http://ls6.informatik.uni-dortmund.de/ir/search/cgi-bin/SFgate
?database=ls6.informatik.uni-dortmund.de%2Fbibdb-html
&database=bib.informatik.uni-dortmund.de%2Fjournals
&%28information+or+retrieval%29+and+%28jahr%3E1995%29

3.1.8.2 Harvest

Harvest (Bowman et. al. (1995)) ist ein Informationssystem zur Sammlung und Indexierung von Internet-Hypertextbeständen und dem Zugriff auf diese Bestände. Es besteht aus den Komponenten gatherer, indexer, broker, replicator und cache.
Als Indexierungs- und Anfragekomponente verwendet Harvest standardmäßig das System Glimpse (Manber, Wu (1994), Manber (1996)), das auf agrep (Manber, Wu (1992)) basiert. Agrep bietet eine eingeschränkte reguläre Suche (außer Zeichenwiederholungen). Es verwendet den 8-Bit-Zeichensatz nach ISO 8859. Anfrageergebnisse werden nicht nach einem Ähnlichkeitsmaß geordnet. Die Syntax der Anfragesprache wird in Camargo (1994) definiert.
Bei Glimpse sind reguläre Ausdrücke aus Effizienzgründen auf eine Länge von 30 Zeichen begrenzt (ohne Metazeichen) und können nur auf einzelne Worte angewendet werden.

Beispielanfragen

Es wird der Harvest Broker: "http://www.tu-berlin.de/harvest-bin/BrokerQuery" mit der Glimpse-Datenbasis: "TUB-alle_WWWs" verwendet (WWW-Bestand von WWW-Servern der TU Berlin im März 1997).

Bei den Harvest-Anfragen (URL) wird für Metazeichen als Fluchtzeichen das Zeichen %, gefolgt von dem hexadezimalen ASCII-Zeichencode benutzt. Das Zeichen " wird z.B. durch die Zeichenfolge %22 ausgedrückt.
Zu jedem Element der Ergebnismenge werden die Felder "Titel" und "URL" und die Volltextzeilen (Ergebnis der regulären Anfrage) angezeigt.

Anfragen nach Dokumenten

1. Welche Dokumente enthalten die Zeichenfolge "information" ?

⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=information
&descflag=on
&opaqueflag=on

als Glimpse-Anfrage:

⇒ glimpse -C -J www.tu-berlin.de -K 8500 -H /home/harvest -l information

2. Welche Dokumente enthalten die Zeichenfolge "information retrieval" (Groß/Kleinschreibung wird nicht unterschieden) ?

⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=%22information%20retrieval%22
&descflag=on
&opaqueflag=on
&caseflag=on

3. Welche Dokumente enthalten die Zeichenfolge "information" oder eine Zeichenfolge, die mit "information" in allen Zeichen außer einem Zeichen (Einfügung, Entfernung oder Modifikation) übereinstimmt ?

⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=information
&descflag=on
&opaqueflag=on
&errorflag=1

4. Welche Dokumente enthalten den regulären Ausdruck "inform.*" (Zeichenfolgen in einer Zeile, die mit "inform" beginnen und beliebig enden) ?

⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=inform%2E%2A
&descflag=on
&opaqueflag=on

5. Welche Dokumente enthalten den regulären Ausdruck "^Fachinformation.*" (Zeichenfolgen in einer Zeile beginnen am Anfang der Zeile mit "Fachinformation" und enden beliebig) ?
⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=%5EInformation%2E%2A
&descflag=on
&opaqueflag=on

6. Welche Dokumente enthalten den regulären Ausdruck "Thesauri$" (Zeichenfolgen in einer Zeile enden mit der Zeichenfolge "Thesauri") ?
⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=Thesauri%24
&descflag=on
&opaqueflag=on

7. Welche Dokumente enthalten den regulären Ausdruck "[Tt]hesaurus.*" (Zeichenfolgen in einer Zeile beginnen mit dem Zeichen "T" oder "t" gefolgt von der Zeichenfolge "hesaurus" und enden beliebig) ?
⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=%5BTt%5Dhesaurus%2E%2A
&descflag=on
&opaqueflag=on

8. Welche Dokumente enthalten den regulären Ausdruck ".*unit\..*" (Zeichenfolgen in einer Zeile, die beliebig beginnen, gefolgt von der Zeichenfolge "unit" und dem Zeichen "." und die beliebig enden) ?
Harvest-Anfrage mit einer WWW-Benutzeroberfläche:
⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=%2E%2Aunit%5C%2E%2E%2A
&descflag=on
&opaqueflag=on

Boolesche Anfragen nach Dokumenten

9. Welche Dokumente enthalten die Zeichenfolgen "information" und "hypertext" ?
⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=information+and+retrieval
&descflag=on
&opaqueflag=on

10. Welche Dokumente enthalten die Zeichenfolgen "information" oder "retrieval"?
⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=information+or+retrieval
&descflag=on
&opaqueflag=on

Anfragen mit Attributen

11. Welche Dokumente enthalten im Attributwert des Attributs "Titel" die Zeichenfolge "Fachinformation" ?

⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=Title%20%3A%20Fachinformation
&descflag=on
&opaqueflag=on

12. Welche Dokumente enthalten im Attributwert des Attributs "Titel" den regulären Ausdruck "[Ii]nformation.*" (Zeichenfolgen, die mit dem Zeichen "I" oder "i" beginnen, gefolgt von der Zeichenfolge "nformation", gefolgt von einer beliebigen Zeichenfolge) ?

⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=Title%3A%5BIi%5Dnformation%2E%2A
&descflag=on
&opaqueflag=on

13. Welche Dokumente enthalten die Zeichenfolgen "Karriere" oder "Geld" und enthalten im Attributwert des Attributs "Titel" die Zeichenfolge "information" ?

⇒ http://www.tu-berlin.de/harvest-bin/BrokerQuery/
?broker=TUB-alle_WWWs
&query=(Karriere+or+Geld)+and+Title%20%3A%20information
&descflag=on
&opaqueflag=on

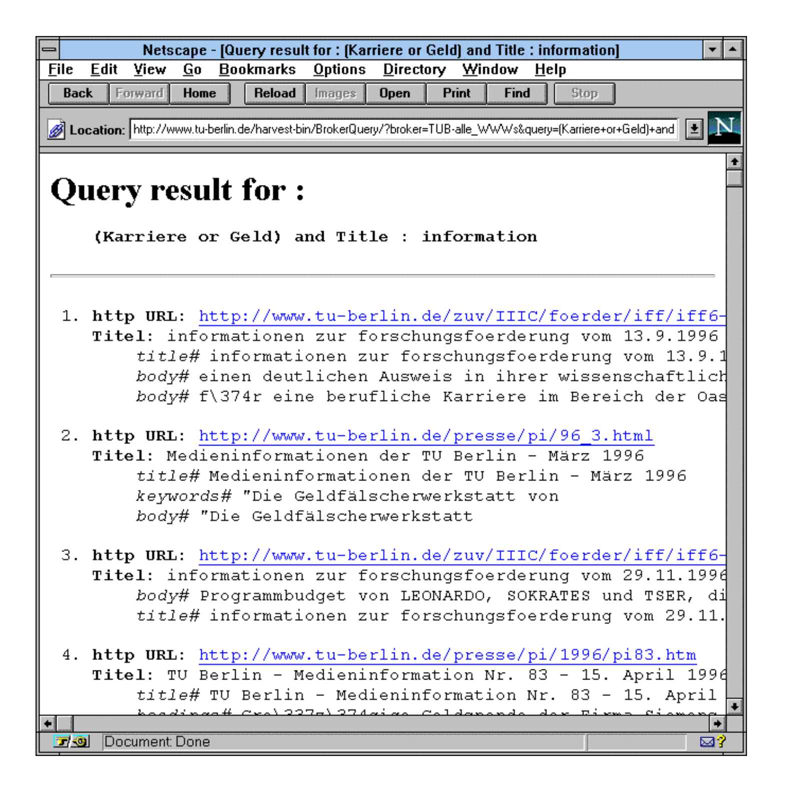


Abbildung 7: Harvest-Anfrage mit Netscape (im März 1997)

3.1.8.3 Suchmaschinen

Es existiert eine Vielzahl von Suchmaschinen wie z.B. Aliweb (2000), AltaVista (2000), Excite (2000), Fireball (2000), Google (2000), Harvest (2000), InfoSeek (2000), Inktomi (2000), Kolibri (2000), Lycos (2000), MetaCrawler (2000), OpenText (1997), Webcrawler (2000) und Yahoo (2000). Eine vergleichende Untersuchung wird in Bekavac (1996), Koch (1995, 1996) und Masermann, Vossen (1998) durchgeführt. Eine Auflistung von Suchmaschinen findet sich in Koster (2001b).

Definition Roboter (vgl. Koster (2001a)): *Roboter sind Programme, die rekursiv von Startdokumenten ausgehend referenzierte Dokumente im Dateisystem ablegen. Dabei ist es unerheblich, welcher Traversierungsalgorithmus oder welche Heuristik verwendet wird. Nachdem alle Startdokumente bearbeitet wurden, werden die Volltextinhalte der Dokumente in invertierte Dateien überführt. Roboter werden auch Spider, Wanderer, Web Crawler oder Worms genannt.*
Definition Suchmaschinen (vgl. Koster (2001a)): *Suchmaschinen sind Suchprogramme für von Robotern aufgebaute Bestände. Suchmaschinen werden auch roboterbasierte Suchdienste genannt.*

Eine Syntax und Semantik der in den Suchmaschinen verwendeten Anfragesprachen wird i.A. nicht bereitgestellt. Bei Einsatz eines Ähnlichkeitsmaßes wird i.A. dessen genaue Definition nicht gegeben, so daß berechnete Rangordnungen der Ergebnisse nicht nachgeprüft werden können.
Es werden folgende Typen von Dokumenten verwendet: XML-Dokument, HTML-Dokument, Newsartikel, E-Mail, FAQ.
In den Suchmaschinen werden folgende Attribute verwendet:
Allgemein: Autor, Inhalt, Schlüsselworte, Zusammenfassung, Zeitraum, Zeitpunkt, Schwellwert für die Relevanz, Dokumenttyp, Sachgebiet
Zusätzlich:
Für XML-Dokumente: eigendefinierte Attribute
Für HTML-Dokumente: URL, Titel, erste Überschrift, Sprache, Rechner (IP-Adresse oder Domain-Name), URL-Verweise, Anzahl der URL-Verweise (abgehend, eingehend), Bild-URL-Adressen, Java-Applets
Für Newsartikel: lokaler Newsserver, Newsgruppen (newsgroups), Thema (subject)
Für E-Mails: Vorname, Nachname, Domainname

In den Suchzeichenfolgen kann als Zeichensatz der 8-Bit-Zeichensatz nach ISO 8859-X verwendet werden. Als Maskierungen können eingesetzt werden: Rechts-, Links- und Innenmaskierung, Wortmaskierung für Phrasen, regulärer Ausdruck, phonetische Ähnlichkeit, Namenerkennung, Synonyme. Die Unterscheidung von Groß/Kleinschreibung kann spezifiziert werden. Weiterhin ist eine Suche mit Attributen (Inhalt des Dokument, alle Attribute oder einzelne Attribute) möglich.

Als boolesche Operatoren werden das "logische und" ("and"), das "logische oder" ("or"), das "logische nicht" ("not") und das "logische binäre nicht" ("and not") unterschieden. Boolesche Ausdrücke können geschachtelt werden. Bei booleschen Anfragen wird standardmäßig keine Ordnung der Ergebnismenge durchgeführt. Bei AltaVista (AltaVista (1997), Ray, Ray, Seltzer (1997)) kann diese ungeordnete Ergebnismenge durch zusätzliche Angabe von Suchbegriffen (Anordnung wie beim Anhäufungsoperator s.u.) geordnet werden.

Wortabstandsoperatoren sind: near und adjacent. Als Standard ist meist der maximale Wortabstand/Zeichenabstand zwischen den Suchzeichenfolgen vorgegeben, ist aber durch eine Zahl spezifizierbar.
Boolesche Operatoren und Wortabstandsoperatoren können verknüpft werden.
Zur Anordnung der Ergebnismengen können folgende Operatoren verwendet werden:

1. Anhäufungsoperator (" "), zweistellig, ähnlich dem "logischen oder": Mit dem Anhäufungsoperator verknüpfte Suchbegriffe (Einzelwort oder Phrase) liefern höhere Anordnungen von Ergebnis-Dokumenten, je mehr von diesen Suchbegriffen und je häufiger diese Suchbegriffe in den Dokumenten vorkommen. Ein Suchbegriff muß mindestens einmal vorkommen. Eine genaue Definition der Ähnlichkeitsfunktion wird von den Suchmaschinenherstellern meist leider nicht veröffentlicht.
2. Plus-Operator ("+"), einstellig: Der Plus-Operator, angewendet auf einen Suchbegriff (Einzelwort oder Phrase), liefert höhere Anordnungen von Ergebnis-Dokumenten, je häufiger dieser in den Dokumenten vorkommt. Voraussetzung ist, daß dieser Suchbegriff mindestens einmal vorkommen muß.
3. Minus-Operator ("-"), einstellig: Der Minus-Operator, angewendet auf einen Suchbegriff (Einzelwort oder Phrase), liefert Ergebnis-Dokumente, in denen dieser nicht in den Dokumenten vorkommt.

Boolesche Operatoren und Operatoren zur Anordnung können nicht verknüpft werden.
Für das Suchergebnis kann spezifiziert werden, welche Felder dargestellt werden sollen und nach welchem Kriterium das Ergebnis geordnet werden soll (z.B. zeitlich nach dem Datum, alphabetisch nach dem Domainnamen etc.).

Beispielanfragen

Anfragen nach Dokumenten
1. Welche Dokumente im globalen Raum der HTML-Dokumente enthalten im Attribut "Inhalt" eine Zeichenfolge, die mit "sing" beginnt ?
Altavista-URL:
⇒ http://altavista.digital.com/cgi-bin/query?pg=q&what=web&fmt=d&q=sing*

Anfragen mit Volltext
2. Welche Dokumente im globalen HTML-Bestand enthalten im Attribut "Inhalt" die Zeichenfolge "James", die einen Wortabstand von höchstens 2 Worten zu der Zeichenfolge "Kirk" hat ?
Webcrawler-URL:
⇒ http://webcrawler.com/cgi-bin/WebQuery?fromhome=0&text=James+near%2F2+Kirk

Anfragen mit Attributen
3. Welche Dokumente im globalen HTML-Bestand verweisen auf das Dokument mit der URL "http://www.cs.tu-berlin.de/" ?
AltaVista-Anfrage (Advanced Search):
⇒ link:http://www.cs.tu-berlin.de
Altavista-URL:
⇒ http://altavista.digital.com/cgi-bin/query?pg=aq&what=web&fmt=.
&q=link%3Ahttp%3A%2F%2Fwww.cs.tu-berlin.de&r=&d0=&d1=

4. Wieviele Dokumente im globalen HTML-Bestand verweisen auf das Dokument mit der URL "http://www.cs.tu-berlin.de/" ?

AltaVista-Anfrage (Advanced Search):
⇒ link:http://www.cs.tu-berlin.de
⇒ as a count only
Altavista-URL:
⇒ http://altavista.digital.com/cgi-bin/query?pg=aq&what=web&fmt=n&
q=link%3Ahttp%3A%2F%2Fwww.cs.tu-berlin.de&r=&d0=&d1=

5. Welche Dokumente im globalen News-Bestand enthalten im Attribut "subject" die Zeichenfolge "Biete Job" ?
AltaVista-Anfrage (Advanced Search):
⇒ subject:"Biete Job"

Boolesche Anfragen
6. Welche Dokumente im globalen HTML-Bestand enthalten im Attribut "Inhalt" eine Zeichenfolge, die mit "sing" beginnt und die Zeichenfolge "Take That" oder die Zeichenfolge "Madonna" ?
AltaVista-Anfrage (Advanced Search):
⇒ sing* and ("Take That" or Madonna)

7. Wieviele Dokumente im globalen HTML-Bestand enthalten im Attritbut "Inhalt" nicht die Zeichenfolge "Madonna" (einstelliges boolesches not)?
AltaVista-Anfrage (Advanced Search):
⇒ not Madonna
⇒ as a count only

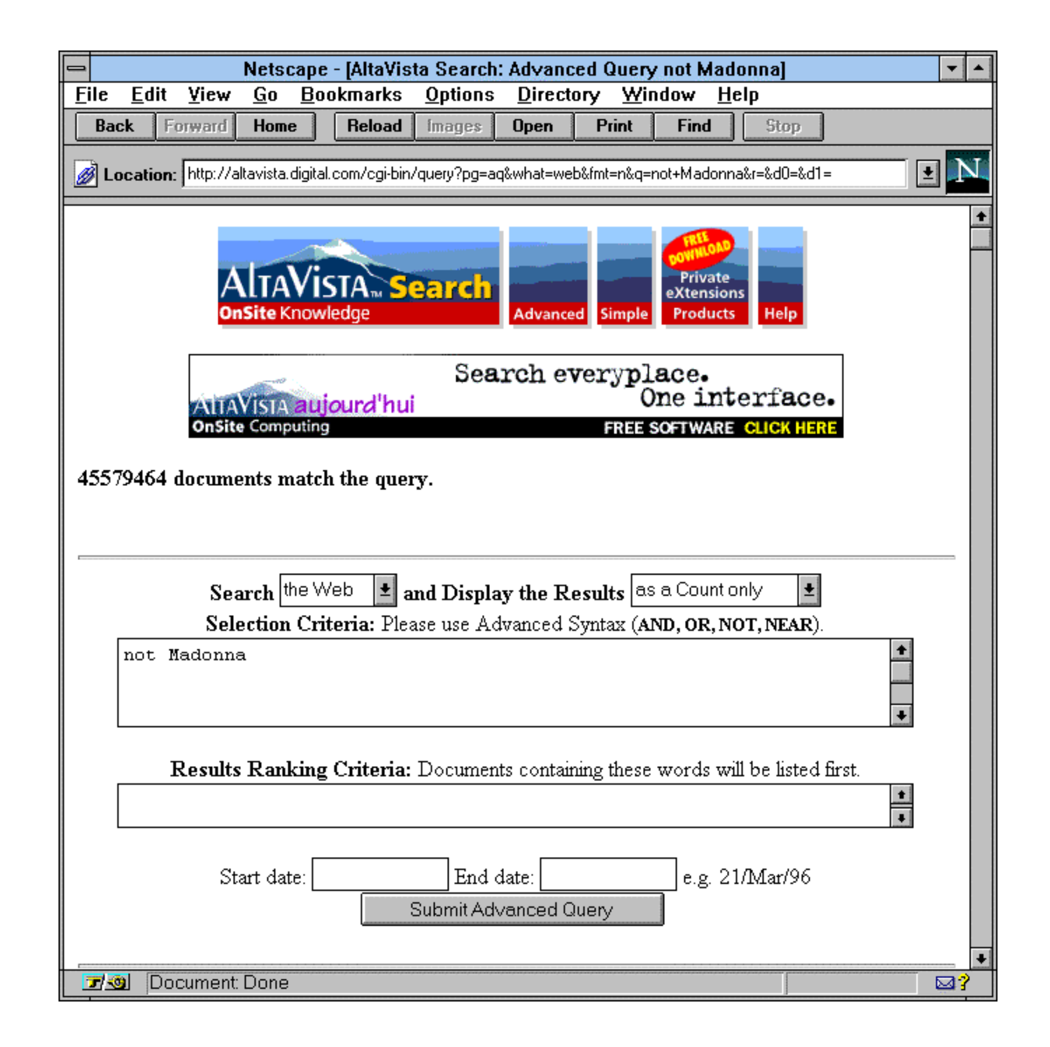


Abbildung 8: AltaVista-Anfrage (im März 1997)

3.2 Suche in link-strukturierten Dokumenten

Im WWW und bei Hyper-G wird eine Zusammenführung der Dienste Telnet, Ftp, News, Gopher, Wais, Http etc. erreicht. Diese Dienste sind jedoch nicht vollständig integriert, so daß Anfragen wie die folgende dort nicht möglich sind: Welche Dokumente vom Typ Ftp-Datei oder vom Typ Html-Dokument enthalten die Zeichenfolge "Compiler".

Anfragen innerhalb der einzelnen Dienste können mit verschiedenen Anwendungssystemen (Clients) gestellt werden. Über eine eindeutige Adresse bzw. Namen des jeweiligen Dienstes (URL, URN) werden Dokumente (Dateien, Verzeichnisse, Artikel, Dokumente, Hypertexteinheiten) identifiziert. Ergebnismengen werden nach unterschiedlichen Kriterien wie zum Beispiel nach dem Datum (bei der Newssuche) oder nach der Anzahl der Treffer (bei der Volltextsuche) geordnet.

Datenbankbasierte Anfragesysteme wie W3QS und WebSQL bieten einen mächtigeren Zugriff auf Hypertextbestände an. Dort werden Netzwerkoperatoren für den Zugriff auf die Verweisstruktur im Hypertextinformationssystemen bereitgestellt. Weiterhin ist es möglich, eine reguläre Volltextsuche durchzuführen.

Folgende Hypertextsuchsysteme werden nicht weiter im Detail vorgestellt:

ARANEUS (Atzeni, Masci et.al. (1998), Araneus (2001)) ist ein Projekt, in dem ein datenbankbasiertes System den Zugriff auf Web-Sites gestattet. Clever (Kleinberg, Kumar et.al. (1999), Kumar, Raghavan et.al. (1999), Clever (2001)) ist ein Projekt, in dem das WWW als ein Graph betrachtet wird. In Clever werden beispielsweise Algorithmen entwickelt, die

“Autoritäten“ (Dokumente, die in Themenlisten aufgeführt werden) und “Themenlisten“ (Dokumente, die wiederum Autoritäten zitieren) im WWW erkennen. WebLog (Lakshmanan, Sadri, Subramanian (1996)) ist eine logische Anfragesprache.

3.2.1 HTTP/URL

Das Hypertext Transfer Protokoll (HTTP) ist ein zustandsloses Protokoll für ein verteiltes Hypertext Informationssystem (RFC 1945). Für jede Anfrage werden vier Operationen in folgender Reihenfolge durchgeführt: 1. Verbindungsaufbau durch den Client, 2. Anfrage durch den Client, 3. Antwort durch den Server und 4. Verbindungsabbbau durch den Server oder den Client (durch Abbruch).
Folgende Anfragen sind in HTTP Version 1.0 möglich:
1. "get": Anfrage nach einem WWW-Dokument unter der angegebenen URL
2. "head": Anfrage nach dem HTTP-Kopf des WWW-Dokuments der angegebenen URL
3. "put": Speichern der im Datenteil gesendeten Daten unter der angebenen URL
4. "post": Übergabe der im Datenteil gesendeten Daten an das unter der URL angegebene Programm (Common Gateway Interface (CGI))
Durch den URL-Mechanismus (RFC 1738) können bisherige Anfragemöglichkeiten im Internet nachgebildet werden. Beispielsweise werden die Protokolle ftp, http, gopher, mailto, news, nntp, telnet, wais, file und prospero für URL-Anfragen zur Verfügung gestellt. Weiterhin bietet der URL-Mechanismus die Möglichkeit, Parameter zu übergeben. Ergebnis einer URL-Anfrage ist genau ein WWW-Dokument.

Mit HTTP können Dokumente unter ihrer spezifizierten URL angefordert werden. Durch Übergabe von Daten an eine CGI-URL können andere Informationssysteme dynamisch angesprochen werden. Beispielsweise ist durch CGI eine Verbindung zu relationalen Datenbanken in Form von SQL-Anfragen möglich.

Beispielanfrage

1. Welches Dokument hat die "URL" "http://www.cs.tu-berlin.de/" ?
HTTP-URL:
⇒ http://www.cs.tu-berlin.de/ ?
Anfragen an andere Informationssysteme (Ftp, Wais, Bibliothekssysteme, Datenbanken etc.) können mit dem URL-Mechanismus über CGI nachgebildet werden.

3.2.2 Hyper-G

Hyper-G ist ein Informationssystem für verteilte Hypertextbestände, das auf dem Client-Server-Modell mit dem Protokoll HG-CSP basiert (Kappe (1993), Andrews, Kappe, Maurer (1995), Dalitz, Heyer (1995)). Der Hyper-G-Server besteht aus den Komponenten: Hyper-G Session Manager (hgserver), objektorientierter Datenbankserver (dbserver), Volltextdatenbankserver (ftserver) und Cache-Server. Hyper-G-Clients wie z.B. Harmony oder Amadeus kommunizieren über HG-CSP mit den Hyper-G-Servern. Im Gegensatz zum WWW arbeitet der Benutzer mit Hyper-G in sogenannten Sitzungen (sessions). Dadurch ist es möglich, Information über das Benutzerverhalten zu sammeln (Statistiken, Benutzerprofile) oder Relevanzfeedbacktechniken einzusetzen.
Mit Hyper-G ist es möglich, Gopher-, WWW- und Hyper-G-Dokumente zu verwalten und auf sie zuzugreifen. An einer FTP- und WAIS-Komponente wird gearbeitet.

Als Dokumentbeschreibungssprache für die Dokumente können HTML (RFC 1866) und HTF (Kappe (1996)) verwendet werden.
Mit Hyper-G können benannte Mengen von Dokumenten, sogenannte Kollektionen, bestimmt werden. Kollektionen sind Mengen von explizit aufgeführten Dokumenten oder wiederum von Kollektionen. Eine Kollektion kann alle Dokumente eines Hyper-G-Servers enthalten (Server-Kollektion). Weiterhin können Kollektionen als Ergebnis einer Anfrage bestimmt werden. Kollektionen können serverübergreifend gebildet werden. Kollektionen können in den Anfragen als Suchräume verwendet werden.
Jedes Dokument besitzt eine Menge von Attributen wie Id, Typ, Name des Dokument, Titel, Besitzer, Zugriffsrechte, Sichtbarkeitsdauer, Autor, Beschreibung, Inhalt, Zeitpunkt der Entstehung, letzter Modifikationszeitpunkt, inhaltliche Deskriptoren, Ordnungsnummer. Kollektionen haben die weiteren Attribute Sortierkriterium und Übersichtsseite.

Beziehungen zwischen Dokumenten:
1. Ausgangs- und Zielregion einer Beziehung kann innerhalb eines Dokuments liegen. Das Dokument kann multimedial sein (Postscript, Bild, Video etc.). Eine Region kann aus mehreren anderen Regionen bestehen.
2. Ausgangs- und Zielregionen von Beziehungen können überlappen.
3. Ausgangs- und Zielregion einer Beziehung kann ein Hyper-G-Dokument (Name, Id), eine Hyper-G-Kollektion oder ein WWW-Dokument (URL) sein.
4. Ausgangs- und Zielregionen einer Beziehung besitzen eine Menge von Attributen: Id, Typ, Zeitpunkt der Entstehung, letzter Modifikationszeitpunkt, lokale Ziel-Id (bzw. Ausgangs-Id), globale Ziel-Id (bzw. Ausgangs-Id), Region (Beginn, Länge).

Darstellungs- und Navigationsmethoden:
Hyper-G bietet fortgeschrittene Darstellungs- und Navigationsmethoden: 3-dimensionale und hierarchische Darstellung von Hypermediabeziehungen und Anfrageergebnissen, Darstellung beschrittener Wege, lokale Karte der eigenen Sitzung.

Anfragen:
Verwendeter Zeichensatz ist ISO 8859-1. Groß- und Kleinschreibung wird in den Anfragen nicht unterschieden. In den Anfragen und Volltextindexen benutzte Zeichenkombinationen wie zum Beispiel ae, oe, ue, ss, oder è werden auf die Einzelzeichen ä, ö, ü, ß und e abgebildet. Stopworte können in den Anfragen nicht benutzt werden. Suchzeichenfolgen können mit dem Zeichen * rechts maskiert werden. Als Attribute werden Suchraum, Titel, Schlüsselworte, Inhalt, Sprache(n), Autor, letzter Modifikationszeitpunkt unterschieden.
Als boolesche Anfrageoperatoren werden das "logische und" (&, &&, and, und), das "logische oder" (| , ‖ , or, oder), das "logische binäre nicht" (&! , andnot, undnicht) und das "Fuzzy-Und" (&[f]) für Volltextanfragen, also Anfragen über den Inhalt von Dokumenten, zur Verfügung gestellt. Ergebnisse von Volltextanfragen werden mit der Hyper-G-Ähnlichkeitsfunktion geordnet. Ergebnis einer "Fuzzy und"-Anfrage ist dieselbe Menge wie das Ergebnis einer "logischen oder"-Anfrage. Diese Menge wird mit dem Hyper-G-Ähnlichkeitsmaß geordnet.
Attribute können nicht in beliebiger boolescher Kombination verknüpft werden. Titel, Schlüsselworte und Inhalt können mit logischem "oder" und die anderen Attribute mit logischen "und" verknüpft werden.

Beispielanfragen

Anfragen nach Hyper-G-Dokumenten
1. Welches Dokument (hier vom Typ Kollektion) enthält im Attribut "URL" die Zeichenfolge "http://hyperg.cs.tu-berlin.de:80/CDB14ED3/Ccs.lehre.vv.ws9596" ?
Hyper-G-URL:
⇒ http://hyperg.cs.tu-berlin.de:80/CDB14ED3/Ccs.lehre.vv.ws9596 ?

Anfragen mit Attributen
2. Welche Dokumente im Bestand "Hyper-G-Collection: Vorlesungsverzeichnis der TU Berlin", haben den Autor "wosch@hyperg.cs.tu-berlin.de" ?
Hyper-G-URL:
⇒ http://hyperg.cs.tu-berlin.de:80/43B01FCA/search.html?ref=cs.lehre.vv.ws9596
Search for Author: ⇒ wosch

3. Welche Dokumente im Bestand "Hyper-G-Collection: Vorlesungsverzeichnis der TU Berlin" enthalten im Attribut "Titel" oder "Inhalt" die Zeichenfolge "Informatik" ?
Hyper-G-URL:
⇒ http://hyperg.cs.tu-berlin.de:80/43B01FCA/search.html?ref=cs.lehre.vv.ws9596
Search for: ⇒ Informatik

Boolesche Anfragen
4. Welche Dokumente im Bestand "Hyper-G-Server des ZIB-Berlin" enthalten im Attribut "Inhalt" die Zeichenfolge "Information" und die Zeichenfolge "Retrieval" oder "Brok*" ?
Hyper-G-URL:
⇒ http://hyperg.cs.tu-berlin.de:80/43B01FCA/search.html?ref=cs.lehre.vv.ws9596
Search for: ⇒ Information & (Retrieval | Brok*)

5. Welche Dokumente im Bestand "weltweit alle Hyper-G-Kollektionen" enthalten im Attribut "Inhalt" die Zeichenfolge "Information" und nach dem Hyper-G-Fuzzymaß berechnet eine ähnliche Zeichenfolge zu "Retrieval" ?
Hyper-G-URL:
⇒ http://hyperg.cs.tu-berlin.de:80/43B01FCA/search.html?ref=cs.lehre.vv.ws9596
Search for: ⇒ Information &[f] Retrieval

3.2.3 W3QS

W3QS ist ein Informationssystem, das den netzartigen Zugriff auf Hypertextbestände mit einer SQL-ähnlichen Anfragesprache (W3QL) gestattet (Konopnicki, Shmueli (1998), W3QS (1999)). Weiterhin kann mit regulären Ausdrücken eine Volltextsuche durchgeführt werden.
W3QS ist über WWW-Benutzeroberflächen oder mit einer Programmierschnittstelle (API) zugreifbar. In der WWW-Benutzeroberfläche können Anfragen benannt werden und Zwischenergebnisse festgehalten werden:



Abbildung 9: WWW-Benutzeroberfläche von W3QS (im März 1999)

Beispielanfragen

1. Welche Dokumente, ausgehend vom Dokument mit der URL "http://www.berlin-tour.de/", enthalten im Attribut "Titel" die Zeichenfolge “Charlottenburg“ ?

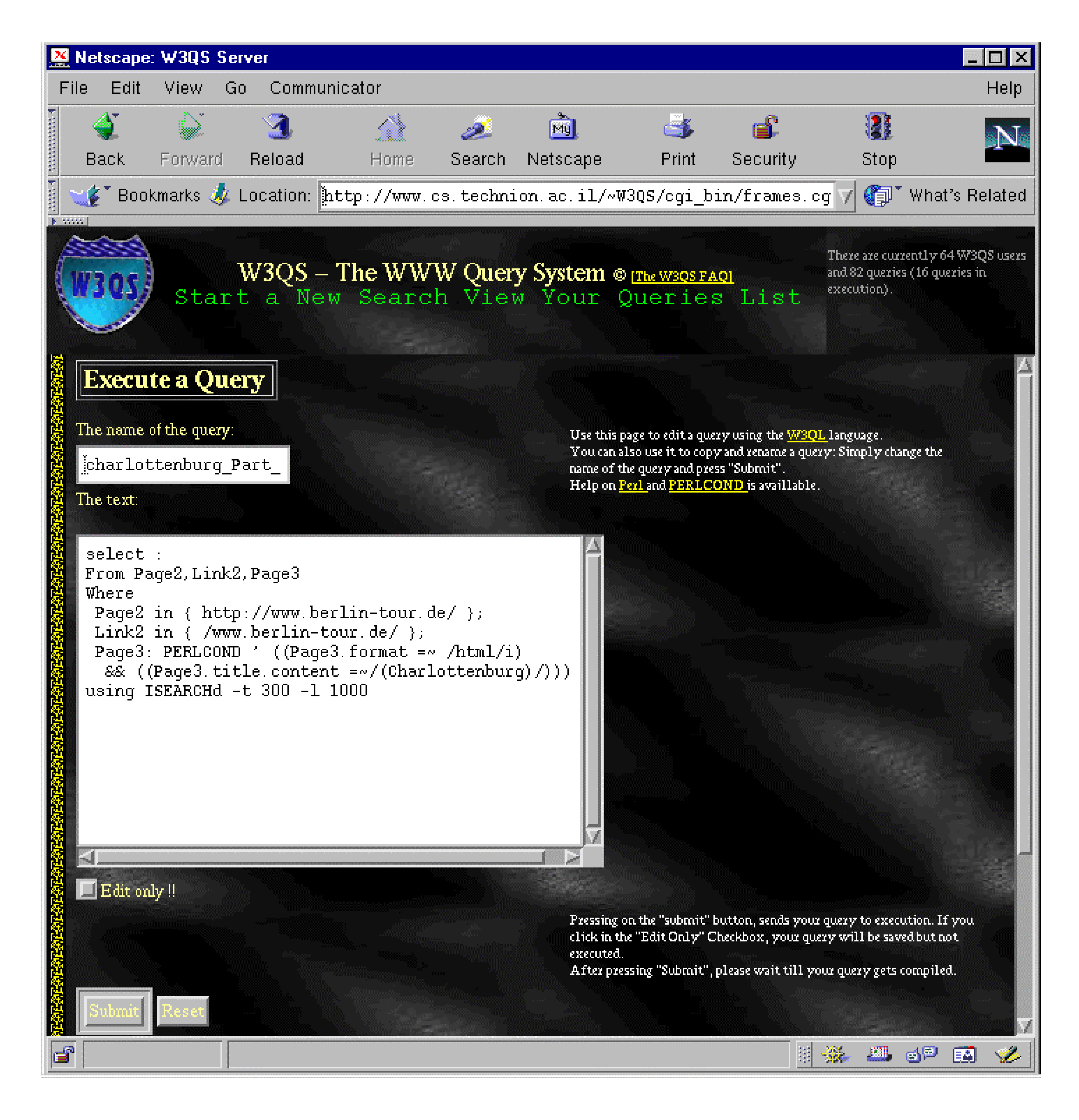


Abbildung 10: W3QS Anfrage (im März 1999)



Abbildung 11: W3QS-Anfrageergebnis (im März 1999)

2. Welche Dokumente vom Typ “Bild“ auf dem Server “http://www.cs.technion.ac.il/“, ausgehend vom Dokument mit der URL “http://www.cs.technion.ac.il/“, sind erreichbar? Aus Performancegründen bleibt die Suchtiefe auf 5 und die Zahl der zu untersuchenden Verbindungen auf 1000 beschränkt.

```
SELECT
FROM n1, I1, (n2, I2), I3, n3
WHERE
        n1 in {http://www.cs.technion.ac.il/};
        I1 in {/cs\.technion\.ac\.il/};
        I2 in {/cs\.technion\.ac\.il/};
        n3: PERLCOND ‘n3.format = ~ /image/‘;
        n3 in {/cs\.technion\.ac\.il/};
        Using ISEARCHd –d 5 –I 1000
```

3. Welche Dokumente, ausgehend vom Dokument “http://wwwwbs.cs.tu-berlin.de/“, enthalten die Zeichenfolge “Schaf“ und sind vom Autor “Willenborg, Josef“ ? Aus Performancegründen bleibt die Suchtiefe auf 4 beschränkt.

```
SELECT
FROM n1, I1, (n2, I2), I3, n3
WHERE
        n1 in {http:// wwwwbs.cs.tu-berlin.de/};
        n3: PERLCOND ‘(n3.content = ~/Schaf/i) && (n3.author = ~/Willenborg\, Josef/i)‘;
        Using ISEARCHd –d 4
```

4. Welche Dateien des Typs “Postscript“ handeln von “Schaf“ (über die cgi-Form “http://lycospro.lycos.com/lycospro-nojava.html“)

```
SELECT n3
FROM n1, I1, (n2, I2), I3, n3
WHERE
        n1 in {http://lycospro.lycos.com/lycospro-nojava.html};
        Run learnform n1 cs76:0 If n1 Unknown in LycosDof;
        Fill I1 In LycosDof with query = “Schaf“
        I1: PERLCOND ‘I1.content = ~/^FORM/i)
        N3: PERLCOND ‘n3.format = ~/postscript/i) && (n3.content = ~/Schaf/i)‘;
        Using ISEARCHd –1 500 –d 3
```

5. Welche fehlerhaften Verbindungen, ausgehend vom Dokument mit der URL “http://www.cs.technion.ac.il/~konop“, existieren. Aus Performancegründen bleibt die Suchtiefe auf 2 beschränkt und wird einmal pro Woche ausgeführt.

```
SELECT
FROM n1, I1, (n2, I2), I3, n3
WHERE
        n1 in { http://www.cs.technion.ac.il/~konop };
        N3: PERLCOND ‘n3.title.content = ~/404 Not Found/i)‘;
        I1 in {/~konop/};
        I2 in {/~konop/};
        Using ISEARCHd –d 2
        Evaluated every week
```

3.2.4 WebSQL

WebSQL ist ein Informationssystem, das den netzartigen Zugriff auf Hypertextbestände mit einer SQL-ähnlichen Anfragesprache gestattet (Arocena, Mendelzon, Mihaila (1997), Mendelzon, Mihaila, Milo (1997), Mendelzon, Mihaila (1997), WebSQL (1999)). Hypertextbestände werden als relationale Datenbank mit zwei Relationen betrachtet:

Document					
url	title	text	length	type	modif

Connection		
base	label	href

Der Zugriff auf Hypertextbestände wird mit Pfadoperatoren ermöglicht. Beispielsweise werden folgende Operatoren bereitgestellt: globaler Link (=>), lokaler Link
(->), interner Link innerhalb eines Dokument: (#>), lokaler Pfad der Länge 0 (=),
lokaler Pfad beliebiger Länge (->*).
Pfade können mit regulären Ausdrücken so eingeschränkt werden, daß Pfade einer (eigendefinierten) Beziehungsart betrachtet werden.

Es ist momentan nicht möglich, das globale Web effizient in einer relationalen Datenbank abzubilden. Deshalb wird mit WebSQL eine Architektur gewählt, die eine Teilmenge des globalen Web betrachtet (z.B. mit Hilfe von Suchmaschinen).

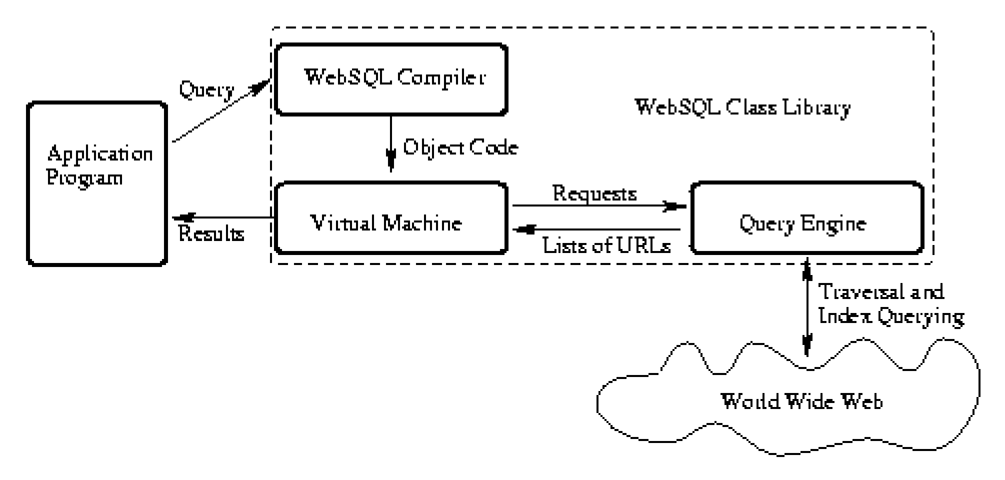


Abbildung 12: Architektur WebSQL (aus Arocena, Mendelzon, Mihaila (1997))

Der WebSQL-Compiler, die Virtual Machine und die Query Engine sind in der Programmiersprache Java implementiert. Benutzer greifen über WWW- oder Java-Benutzeroberflächen auf die zentralen Server zu.

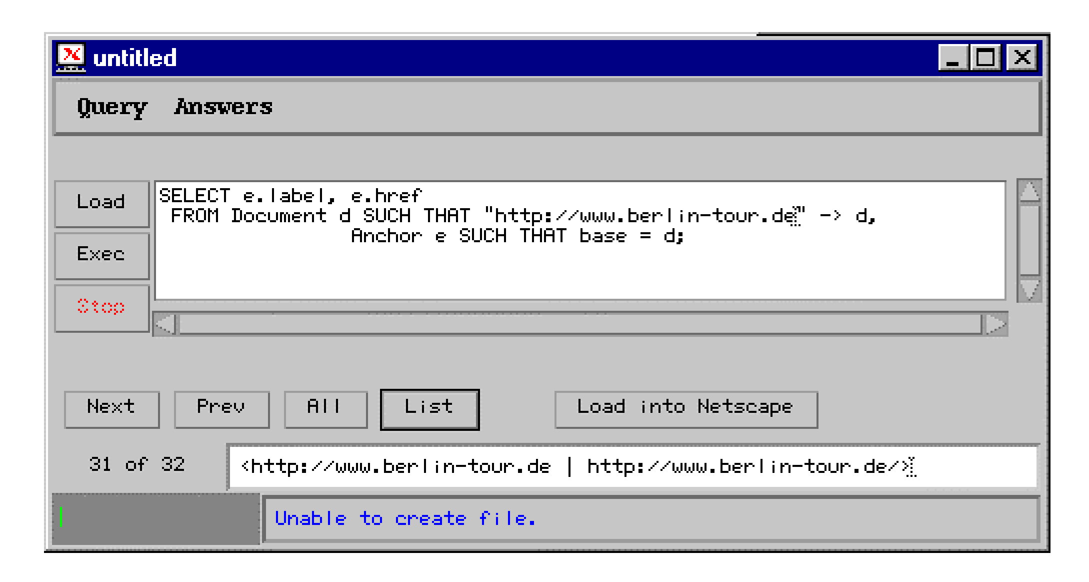


Abbildung 13: WebSQL-Anfrage mit der Java-Benutzeroberfläche (im März 1999)

```
untitled
<Deutsch erh ltlich | http://www.berlin-tour.de/>
<espa ol | http://www.berlin-tour.de/span/>
<Francais/Italiano | http://www.berlin-tour.de/eng/edda.html>
<General city tours | http://www.berlin-tour.de/eng/general.html>
<Berlin - the new capital | http://www.berlin-tour.de/eng/architecture.html>
<Charlottenburg Palace (New Wing) | http://www.berlin-tour.de/eng/palace.html>
<Potsdam | http://www.berlin-tour.de/eng/potsdam.html>
<Further tours on request | http://www.berlin-tour.de/eng/furtherTours.html>
<e-mail | mailto:christ@berlin-tour.de>
<phone, fax or write | http://www.berlin-tour.de/eng/address.html>
<English | http://www.berlin-tour.de/eng/>
<Deutsch erh ltlich | http://www.berlin-tour.de/>
<Francais/Italiano | http://www.berlin-tour.de/span/edda.html>
<tour a pie | http://www.berlin-tour.de/span/apie.html>
<Visita general de la ciudad | http://www.berlin-tour.de/span/general.html>
<Berl n - la nueva capital | http://www.berlin-tour.de/span/capital.html>
<Palacio Charlottemburgo (Ala Nueva) | http://www.berlin-tour.de/span/palacio.html>
<Potsdam | http://www.berlin-tour.de/span/potsdam.html>
<Visitas de la ciudad a petici n | http://www.berlin-tour.de/span/otrasTemas.html>
<llamarme, escribir, mandarme un fax | http://www.berlin-tour.de/span/direccion.html>
< e-mail | mailto:christ@berlin-tour.de>
<back | http://www.berlin-tour.de/>
Dismiss
```

Abbildung 14: WebSQL-Anfrageergebnis mit der Java-Benutzeroberfläche (im März 1999)

Beispielanfragen

1. Welche Paare von URL von Dokumenten haben den gleichen Titel ?

SELECT	d1.url, d2.url
FROM	Document d1, Document d2
WHERE	d1.title = d2.title ANDNOT (d1.url = d2.url)

2. Welche Paare von URL von Dokumenten, die von "Charlottenburg" handeln, haben den gleichen Titel ?

SELECT	d1.url, d2.url
FROM	Document d1 SUCH THAT d1 MENTIONS "Charlottenburg", Document d2 SUCH THAT d2 MENTIONS "Charlottenburg"
WHERE	d1.title = d2.title ANDNOT (d1.url = d2.url)

3. Welche Dokumente, ausgehend vom Dokument mit der URL "http://www.berlin-tour.de" (bel. Suchtiefe), enthalten in Ihrem Titel oder Inhalt die Zeichenfolge "Charlottenburg" ?

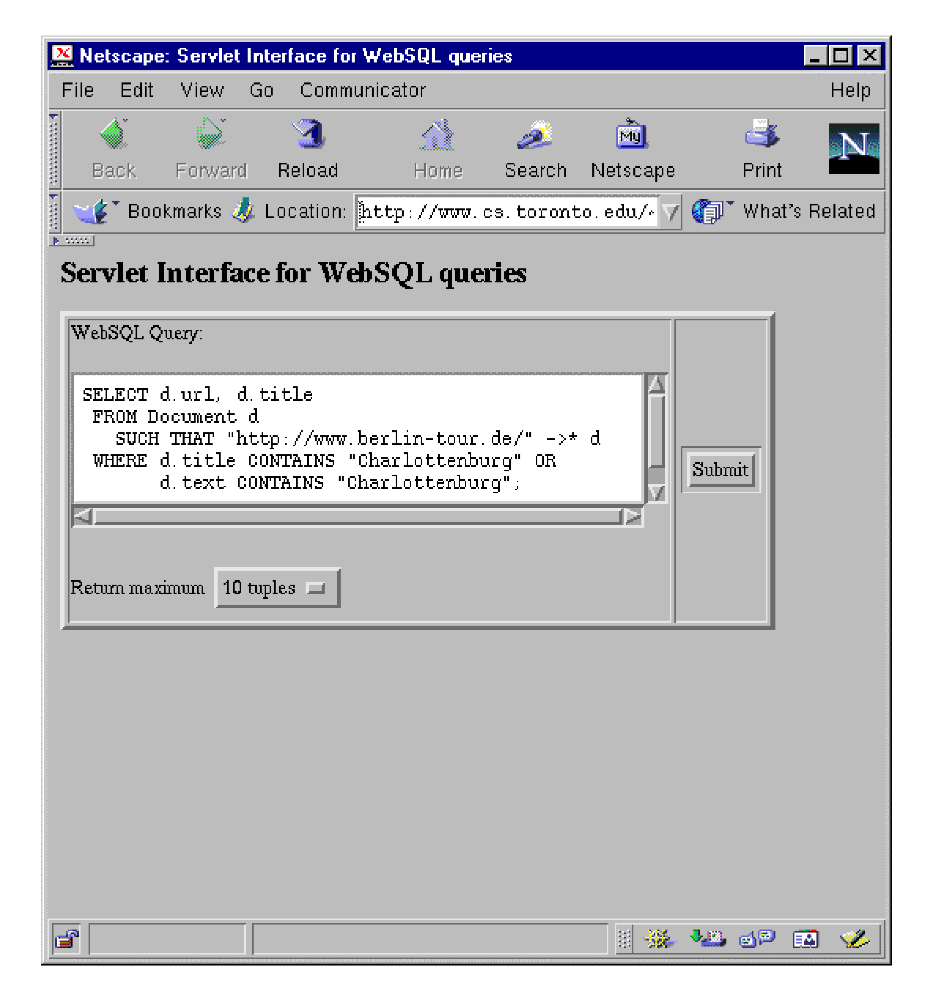


Abbildung 15: WebSQL-Anfrage mit der WWW-Benutzeroberfläche (im März 1999)

4. Welche Dokumente, ausgehend vom Dokument mit der URL "http://www.berlin-tour.de" (bel. Suchtiefe), mit der Beziehungsart content-link" enthalten in ihrem Titel oder Inhalt die Zeichenfolge "Charlottenburg" ?

```
SELECT    d.url, d.title
FROM      Document d SUCH THAT "http://www.berlin-tour.de" [content-link]* d
WHERE     d1.title CONTAINS "Charlottenburg"
                OR d.text CONTAINS "Charlottenburg"
```

5. Welche Beziehungsart existiert zwischen Dokumenten vom Dokument mit der URL "http://www.berlin-tour.de" zu Dokumenten, deren URL die Zeichenfolge ".gif" enhält ?

```
SELECT    a.label
FROM Anchor a SUCH THAT base = "http://www.berlin-tour.de"
WHERE a.href CONTAINS  ".gif"
```

3.3 Suche in geschachtelten Dokumenten

3.3.1 TSIMMIS

TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) ist ein Projekt, das es sich als Ziel gesetzt hat, Informationsbestände, die geschachtelte Dokumente enthalten, einheitlich zur Verfügung zu stellen (Quass, Rajaraman, et.al. (1995), Papakonstantinou, Garcia-Molina, Widom (1995)). In TSIMMIS wurde die objektorientierte Repräsentationssprache

OEM entwickelt, die als Zwischensprache für die Integration der einzelnen Informationsbestände fungiert. Clients greifen mit der Anfragesprache OEM-QL einheitlich auf die Bestände zu. Zur Kommunikation wird das CORBA-basierte DLIOP (Garcia-Molina, Paepcke (1996)) verwendet, das in der Digital Library Initiative im Einsatz ist. DLIOP ist nicht zustandslos, so daß Zwischenergebnisse weiterverwendet werden können.

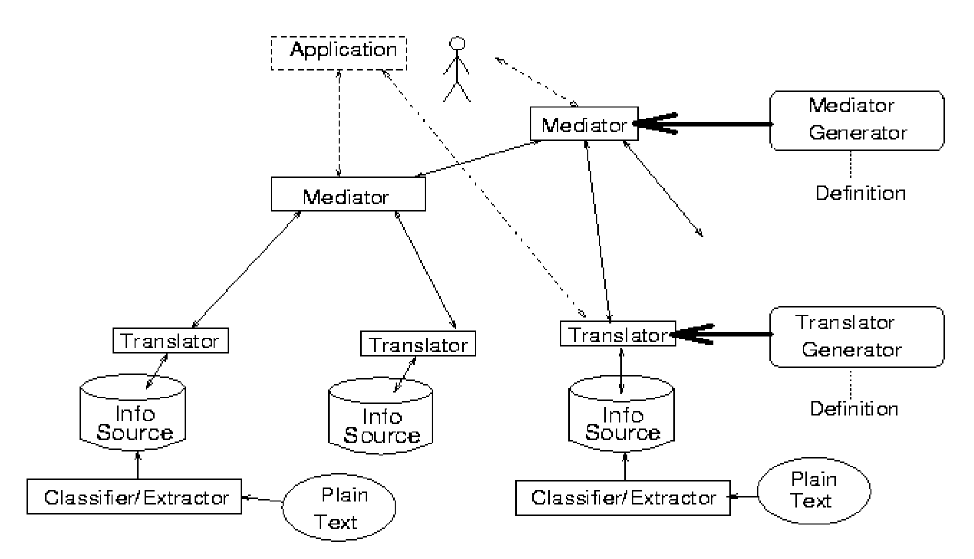


Abbildung 16: TSIMMIS-Architektur (aus Garcia-Molina, Hammert et.al. (1995))

OEM-QL verwendet Mediatoren für Informationsbestände. Mediatoren sind Wissensbasen, in denen Wissen über eine oder mehrere Informationsbestände gespeichert wird. Mit den Mediatoren wird die OEM-QL-Anfrage aufbereitet und an weitere Mediatoren oder Anfrageübersetzer weitergeleitet. Anfrageübersetzer übersetzen die Anfrage in die Zielsprache und leiten sie an das Zielsystem weiter. Anfrageergebnisse werden über den Anfrageübersetzer und den Mediatoren zum Client zurückgegeben.
Anwendungsbereich für TSIMMIS sind Bibliotheken aber auch WWW- und andere Internet-Informationssysteme.

Beispielanfragen

1. Welche Themen werden in Dokumenten behandelt, deren Autor “Ullman“ ist ?

```
SELECT   bibliography.document.topic
FROM     root
WHERE    bibliography.document.author-set.author-last-name = “Ullman“
```

2. Welche Themen werden in Dokumenten behandelt, die eine Exemplarnummer haben ?

```
SELECT   bibliography.?.topic
FROM     root
WHERE    bibliography.?.item-no
```

3. Welche Themen werden in Dokumenten behandelt, die eine gegebene Exemplarnummer haben ?

```
SELECT   *.topic
FROM     root
WHERE    *.item-no
```

4. Welche Dokumente sind von den Autoren “Aho“ und “Hopcroft“ ?

```
SELECT    bibliography.document
FROM      root
WHERE     bibliography.document.author-set.author-last-name (a1) = “Aho“  AND
          bibliography.document.author-set.author-last-name (a2) = “Hopcroft“
```

⟨answer, set, {*obj*}⟩
***obj* is ⟨document, set, {$authors_2$, $topic_2$, $call\text{-}number_2$}⟩**
$authors_2$ is ⟨author-set, set, {$author_2^1$, $author_2^2$, $author_2^3$}⟩
$author_2^1$ is ⟨author-last-name, string, “Aho”⟩
$author_2^2$ is ⟨author-last-name, string, “Hopcroft”⟩
$author_2^3$ is ⟨author-last-name, string, “Ullman”⟩
$topic_2$ is ⟨topic, string, “Algorithms”⟩
$call\text{-}no_2$ is ⟨dewey-decimal, string, “BR273”⟩ □

Abbildung 17: Anfrageergebnis als OEM-Objekt (aus Papakonstantinou, Garcia-Molina, Widom (1995))

3.3.2 XQL

XML ist eine Teilmenge von SGML (Goldfarb, Prescod (2001), ISO 8879). XML-Dokumente bestehen aus Elementen, die wiederum aus Elementen bestehen usw. Elemente können Attribute enthalten.
XQL (Lapp, Robie, Schach (1998)) ist eine Anfragesprache für XML-Dokumente. Eine Anfrage wird an genau ein XML-Dokument gestellt. Ergebnis ist eine Menge von XML-Elementen oder ein XML-Attribut. Die Suchergebnisse können angereichert werden, so daß sie wieder XML-Dokumente darstellen. An diese können dann erneut XQL-Anfragen gestellt werden.
XQL enthält:

1. Operatoren für Teile (XML-Elemente) eines XML-Dokuments: Teile der Stufe n (“/ “), alle Teile (“//“)
2. Einen Operator für die Stelle im XML-Dokument (“ [n] “)
3. Vergleichsoperatoren für Werte von XML-Elementen oder XML-Attributen: Gleichheit, Ungleichheit, kleiner als, größer als, usw.
4. Boolesche Operatoren: logisches und, logisches oder, logisches unäres nicht, logisches binäres nicht
5. Quantoren: es existiert ein, für alle
6. Mengenoperatoren: Vereinigung, Schnittmenge

Beispielanfragen

Alle Anfragen beziehen sich auf ein beliebiges aber festes XML-Dokument.

1. Welche “author“-Elemente existieren ?
⇒ //author

2. Welche “first-name“-Elemente sind direkte Teile der “author“-Elemente ?
⇒ author/first-name

3. Welche “title“-Elemente sind Teile der “bookstore“-Elemente ?
⇒ bookstore//title

4. Welche “last-name“-Elemente sind Teile der 2. Stufe des “book“-Elements ?
⇒ book/*/last-name

5. Welche Elemente enthalten das Attribut “speciality“ ?
⇒ *[@speciality]

6. Welche “title“-Elemente sind direkte Teile von “book“-Elementen, die direkt mindestens ein “excerpt“-Element enthalten ?
⇒ book[excerpt]/title

7. Welche “author“-Elemente enthalten direkt “last-name“-Elemente dessen Wert “Bob“ ist ?
⇒ author[last-name = ‘Bob‘]

8. Welche “author“-Elemente enthalten direkt “last-name“-Elemente, dessen Wert der gleiche ist wie die “guest“-Elemente, die direkt “last-name“-Elemente enthalten ?
⇒ author[last-name = /guest/last-name]

9. Welches ist das “author“-Element an dritter Stelle, das direkt “first-name“-Elemente enthält ?
⇒ author[first-name] [2]

10. Welche Elemente mit dem Suffix-Namen “book“ und dem Prefix-Namen “my“ enthalten direkt ein “author“-Element ?
⇒ my:book[author]

11. Welche “exchange“-Attribute besitzen die “price“-Elemente ?
⇒ price/@exchange

12. Welche “author“-Elemente enthalten direkt mindestens ein “degree“-Element oder ein “award“-Element und mindestens ein “publication“-Element ?
⇒ author[(degree or award) and publication]

13. Welche “author“-Elemente enthalten direkt “last-name“-Elemente, dessen Wert “Bob“ ist und “price“-Elemente dessen Wert größer als “50“ ist ?
⇒ author[last-name = ‘Bob‘ and price gt 50]

14. Welche “first-name“-Elemente und “last-name“-Elemente existieren ?
⇒ first-name $union$ last-name

15. Welche “author“-Elemente enthalten direkt “last-name“-Elemente, deren Werte alle nicht “Bob“ sind ?
⇒ author[all last-name != ‘Bob‘]

3.3.3 Intermedia

Intermedia (Oracle (2000a), Oracle (2000b)) ist ein Informationssystem, das die Speicherung von und den Zugriff auf Volltexte und multimediale Dokumente ermöglicht. Intermedia ist eine Teilkomponente des Datenbanksystems Oracle und liegt uns in der Version 8.16 vor. Es wurde insbesondere im letzten Jahr stark ausgebaut und unterstützt seitdem beispielsweise Operatoren für den Zugriff auf geschachtelte Dokumente.

In den Suchzeichenfolgen kann als Zeichensatz Unicode (UTF8) oder der 8-Bit-Zeichensatz nach ISO 8859-X verwendet werden. Als Maskierung wird die Rechts-, Links- und Innenmaskierung angeboten (Mehrzeichenmaskierung mit dem Zeichen "%", Einzeichenmaskierung mit dem Zeichen "_"). Weiterhin können Operatoren zur Bestimmung linguistisch ähnlicher Worte verwendet werden:

1. Phonetische Ähnlichkeit (Operator "!"): momentan für den 7 Bit Zeichensatz (ASCII) und etwas ineffizienter für ISO 8859-X. Beispielsweise würde "!smith" erweitert zu smit und smythe.
2. Vollformenerweiterung (Operator "$"): momentan für die Sprachen Englisch, Deutsch, Französisch, Spanisch, Italienisch und Holländisch. Beispielsweise würde "$buy" erweitert zu buys, bought, ... aber nicht zu buyer.
3. Rechtschreibungsähnlichkeit (Operator "?"): momentan für den 7 Bit Zeichensatz (ASCII) und etwas ineffizienter für ISO 8859-X. Beispielsweise würde "?cat" erweitert zu cats, calc, case).

Die Unterscheidung von Groß/Kleinschreibung kann in den Suchanfragen spezifiziert werden. Stopwortlisten können angegeben werden. Stopwörter werden nicht in den Index aufgenommen.

Für die Berechnung der Ähnlichkeit von Anfrage und Dokument wird eine Definition in Form eines Algorithmus ("score algorithm") gegeben, so daß die Rangordnungen der Ergebnisse geprüft werden können. Angeordnete Ergebnismengen können mit einem Gewicht (threshold) beschränkt werden. Anfrageterme können gewichtet werden. Beispielsweise würde die Anfrage "cat or dog*3" Dokumente, die "dog" enthalten 3 mal höher gewichten und dementsprechend anordnen.

Intermedia bietet Suchattribute für Dokumente an:

HTML-Dokumente: HTML 4.0 konforme Elemente und Metatags

Newsartikel: RFC 1036 konforme Felder

XML-Dokumente: systemdefinierte Attribute

Eigendefinierte Dokumenttypen: eigenbenannte Elemente, die auch überlappen können

Anfragen mit Attributen werden mit dem Operator "within" gestellt. Suchergebnisse werden dabei nicht geordnet. Anfragen mit Attributen können boolesche Operatoren und den Operator "near" enthalten. In der vorliegenden Intermedia-Version 8.16 können Anfragen mit Attributen geschachtelt werden.

Als Wortabstandsoperator wird "near" bereitgestellt. Der maximale Wortabstand zwischen den Suchzeichenfolgen kann durch eine Zahl angegeben werden. (Standard ist 100). Near liefert ein Ähnlichkeitsmaß für die Dokumente zurück. "Near" kann zusammen mit dem Operator "within" verwendet werden.

Als boolesche Operatoren werden das "logische und" ("and"), das "logische oder" ("or"), und das "logische binäre nicht" ("not") unterschieden. Boolesche Ausdrücke können geschachtelt verwendet werden. Normalerweise wird bei den booleschen Operatoren die Ergebnismenge nicht geordnet. Bei Intermedia wird bei den booleschen Anfragen eine Ordnung der Ergebnismenge durchgeführt: das "logische und" von term1 und term2 liefert das Minimum der Ähnlichkeitswerte für term1 und term2 (beide Terme müssen mindestens einmal vorkom

men), das "logische oder" von term1 und term2 liefert das Maximum der Ähnlichkeitswerte für term1 und term2 (ein Term muß mindestens einmal vorkommen) und das "logische binäre nicht" von term1 und term2 liefert den Ähnlichkeitswert von term1.
Der zweistellige Anhäufungs-Operator ("," bzw. "accumulate"), ähnlich dem "logischen oder", beeinflußt die Anordnung der Ergebnismenge: Je mehr Suchbegriffe der Anhäufung in einem Dokument vorkommen, desto höher ist die Anordnung dieses Dokuments in der Ergebnismenge. Mindestens ein Suchbegriff der Anhäufung muß mindestens einmal in jedem Dokument vorkommen. Beispielsweise würden für die Anfrage "dog, cat" Dokumente, in denen das Wort dog 3 mal und das Wort cat 2 mal vorkommt, höher bewertet werden als Dokumente, in denen nur das Wort dog 6 mal vorkommt.
Der zweistellige Minus-Operator("-"), ähnlich dem "logischen binärem nicht" ordnet Dokumente im Ergebnis niedriger an, wenn sie einen Suchbegriff enthalten. Beispielsweise würden für die Anfrage "dog - cat" Dokumente, in denen das Wort "dog" 3 mal und das Wort "cat" 1 mal vorkommt, höher bewertet werden als Dokumente, in denen das Wort "dog" 3 mal und das Wort "cat" 2 mal vorkommt.
Intermedia stellt folgende terminologische Operatoren zur Verfügung:

1. Oberbegriffe/Unterbegriffe: Der Suchbegriff wird um die Unterbegriffe/Oberbegriffe bis zu einer bestimmte Tiefe erweitert. Es können die Thesaurusrelationen BT (broader term), BTG broader term generic), BTP (broader term partitive) und BTI (broader term instance) bzw. NT (narrower term), NTG (narrower term generic), NTP (narrower term partitive) und NTI (narrower term instance) verwendet werden. Die Angabe einer Suchtiefe ist möglich. Falls mehrere Oberbegriffe/Unterbegriffe mit demselben Namen existieren (Homographen), kann angegeben werden, für welche Begriffe die Oberbegriffe/Unterbegriffe bestimmt werden sollen.
2. Verwandte Begriffe: der Suchbegriff wird um verwandte Begriffe erweitert.
3. Synonyme: der Suchbegriff wird um synonyme Begriffe erweitert.
4. Übersetzungen: der Suchbegriff wird um seine Übersetzungen (Angabe der Sprache möglich) erweitert.
5. Synonyme und Übersetzungen: der Suchbegriff wird um seine synonymen Bezeichnungen und deren Übersetzungen (Angabe der Sprache möglich) erweitert.
6. Vorzugsbegriff: anstelle des Suchbegriffs wird der Vorzugsbegriff in der Anfrage verwendet.
7. Wurzelbegriff bzgl. der Oberbegriffsrelation (top term): anstelle des Suchbegriffs wird der oberste Begriff des Suchbegriffs (bzgl. der Oberbegriffsrelation) in der Anfrage verwendet.

Für alle terminologischen Operatoren gilt, daß der Thesaurus, der verwendet werden soll, mit dem Namen angegeben werden kann.
Intermedia bietet eine sogenannte automatische Themengenerierung für die Dokumente an, um die Präzision der Ergebnisse zu verbessern. Mit Hilfe des Operators "about" können dann thematische Anfragen gestellt werden. Momentan ist dies nur in den Sprachen Englisch und Französisch möglich. Die deutsche Sprache soll Mitte bis Ende 2001 unterstütz werden.
Für das Suchergebnis kann spezifiziert werden, welche Felder dargestellt werden sollen und nach welchem Kriterium das Ergebnis geordnet werden soll (z.B. zeitlich nach dem Datum, alphabetisch nach dem Domainnamen).
Um einen Suchraum für Dokumente zu spezifizieren, kann das Dokument um ein Feld ("search area") erweitert werden. Dieses kann dann mit einem Attribut abgefragt werden.

Beispielanfragen

Anfragen nach Dokumenten
1. Welche Dokumente enthalten Zeichenfolgen, die mit "sing" beginnen ?
⇒ sing%

2. Welche Dokumente enthalten die Zeichenfolge "dog", gefolgt von "cat", gefolgt von "mouse", in einem Wortabstand von höchstens 3 Worten ?
⇒ near((dog, cat, mouse), 3, true)

Anfragen mit Attributen
3. Welche Dokumente enthalten im Attribut "title" die Zeichenfolge "dog" ?
⇒ dog within title

4. Welche Dokumente enthalten im Attribut "surname" innerhalb des Attributs "author" die Zeichenfolge "Salton" ?
⇒ (salton within surname) within author

5. Welche Dokumente enthalten im Attribut "title" die Zeichenfolge "dog" mit einem Wortabstand von 5 Worten zu "cat"?
⇒ near((dog, cat), 5) within title

Boolesche Anfragen
6. Welche Dokumente enthalten im Attribut "title" eine Zeichenfolge, die mit "sing" beginnt und die Zeichenfolge "Take That" oder die Zeichenfolge "Madonna" und enthalten im Attribut "url" nicht die Zeichenfolge ".com" ?
⇒ sing% and (Take That or Madonna) within title not %.com within url

Terminologische Anfragen
7. Welche Dokumente enthalten die Zeichenfolge "car" oder Unterbegriffe von "car" bis zur Tiefe 3 in dem Thesaurus "vehicles"?
⇒ nt(car, 3, vehicles)

8. Welche Dokumente enthalten die Zeichenfolge "car" oder Synonyme oder deutsche Übersetzungen von "car" in dem Thesaurus "vehicles" oder einen Wurzelbegriff von "car" in dem Thesaurus "general lexicon" ?
⇒ trsyn(car, german, vehicles) or tt(car, general lexicon)

3.4 Faktensuche

Es existieren Datenbanken in unterschiedlichen Anwendungsbereichen. Häufig verwendet wird die Anfragesprache SQL (ISO 9075). Eine objektorientierte Datenbankanfragesprache ist OQL (Cattell (1994)). Verteilte Anfragen sind möglich.
Als Zeichensatz wird überwiegend der 8-Bit Zeichensatz nach ISO 8859 verwendet. An einer Portierung nach Unicode wird gearbeitet.
Über Schnittstellen (CGI, ODBC, JDBC, WWW-Server-Erweiterungen etc.) werden Datenbanken an das WWW angebunden (Informix (1997), Oracle (2001)). Teilweise wird eine automatische Konvertierung der Anfrageergebnisse nach HTML ermöglicht.

Eine relationale Datenbank für WWW-Dokumente ist RBSE (RBSE (1997)). Das Datenmodell sieht Tabellen für Dokumente, für Verweise in den Dokumenten, für den Index etc. vor.
Große kommerzielle Datenbanken werden mit Dokumentationssystemen im WWW präsentiert. Wichtige Anfragesprachen dafür sind Messenger für STN (STN (2001)) und Grips für DIMDI (DIMDI (2001)) etc. Zur Mächtigkeit dieser Anfragesprachen siehe auch Reiner (1991).
Das System NLightN (NlightN (1997)) bietet ein umfangreiches, kommerzielles Angebot an Datenbanken, Bibliotheken, Buchhandlungen, Lexika, Internetbeständen etc. InfoSeek (InfoSeek (2000)) bietet ein ähnliches aber nicht so umfangreiches Angebot.

Beispielanfragen

1. Welche Verweise hat das Dokument, deren URL "http://rbse.jsc.nasa.gov/" ist ?
RBSE-Anfrage:

⇒ SELECT s.url
FROM SPIDER s, SOURCE r
WHERE s.id = r.id and r.src = "http://rbse.jsc.nasa.gov/"

2. Welche Nachnamen von Personen in den Datenbanken "univerzeichnis" der TU Berlin und der HU Berlin beginnen mit dem Zeichen "K" ?
Informix-Anfrage (vgl. Petkovic (1995)):

⇒ SELECT nachname
FROM univerzeichnis@informix_tu_berlin:person,
univerzeichnis@informix_hu_berlin:person
WHERE nachname LIKE ‘K%’

3. Welche Datensätze enthalten in einer ihrer Spalten die Zeichenfolge "Army" oder die Zeichenfolge "Navy" und nicht die Zeichenfolge "Air Force" (Suchraum: alle NlightN Datenbanken) ?
NlightN-Anfrage:

⇒ (Army | Navy) ^ (Air Force)

3.5 Suche nach terminologischen Einträgen

3.5.1 Anfragen

Eine terminologische Anfrage liefert eine Teilmenge des zugrundeliegenden Terminologie- und Dokumentbestands als Ergebnis zurück. Eine terminologische Anfragesprache ist die Basis für den Zugriff auf Dokumentbestände durch Suchkommandos, Navigations- und Darstellungsmethoden. Suchkommandos bieten für bestimmte Anwendungen einen effizienteren Zugriff als Navigations- und Darstellungsmethoden. Wenn jemand beispielsweise genau weiß, daß er Veröfentlichungen nach 1994 von einem Autor mit dem Nachnamen "Müller" aus dem Themenbereich "Datenbanken" sucht, dann kann er mit präzisen Suchkommandos (unterstützt von einer adäquaten Benutzeroberfläche) effizienter anfragen als mit Navigationsmitteln.
Ansätze für terminologische Anfragesprachen finden sich in:

- Common Command Language (CCL, ISO 8777): Es werden Syntax und Anfragebeispiele gegeben. Beispielsweise basiert die Anfragesprache Grips auf CCL.
- Messenger (2001): Es werden Syntax und Anfragebeispiele gegeben. Messenger wird beispielsweise vom Datenbankanbieter STN eingesetzt.

- IQL-DIT mit Thesaurusoperatoren (Reiner (1991) S. 107): Es werden Syntax und Semantik definiert.
- Blair (1990) S. 55-64: Es werden Syntax und Anfragebeispiele für einfache und gewichtete Thesaurusanfragen gegeben.
- Becavac (1996) S. 200: Es wird die "Fischsuche" mit Anfragebeispielen gegeben und ein System im WWW (Moasic-fish) vorgestellt.
- Hyperkatalog Innsbruck (Niedermair (1995)): Es werden Anfragebeispiele gegeben und ein System im WWW (Hyperkatalog Innsbruck (2001)) bereitgestellt.
- THW^{3}Query (Hartlep (1996)): Es werden Syntax, Semantik und Anfragebeispiele gegeben. Ein System wird im WWW bereitgestellt.
- Cyc (Guha, Lenat (1990a), Guha, Lenat (1990b, Cyc (2001)): Es werden Syntax, Anfragebeispiele und ein System im WWW bereitgestellt.
- WordNet (Miller (1995), WordNet (1997)): Es werden Anfragebeispiele und ein System im WWW bereitgestellt.

Beispielanfragen

Anfragen nach terminologischen Einträgen

1. Welche terminologischen Einträge haben einen Namen, der mit der Zeichenfolge "fahr" beginnt (Groß/Kleinschreibung wird nicht unterschieden, Ergebnis wird alphabetisch geordnet)?
ISO 8777-Anfrage:
⇒ scan ct=fahr

2. Welche terminologischen Einträge enthalten den regulären Ausdruck car.* ?

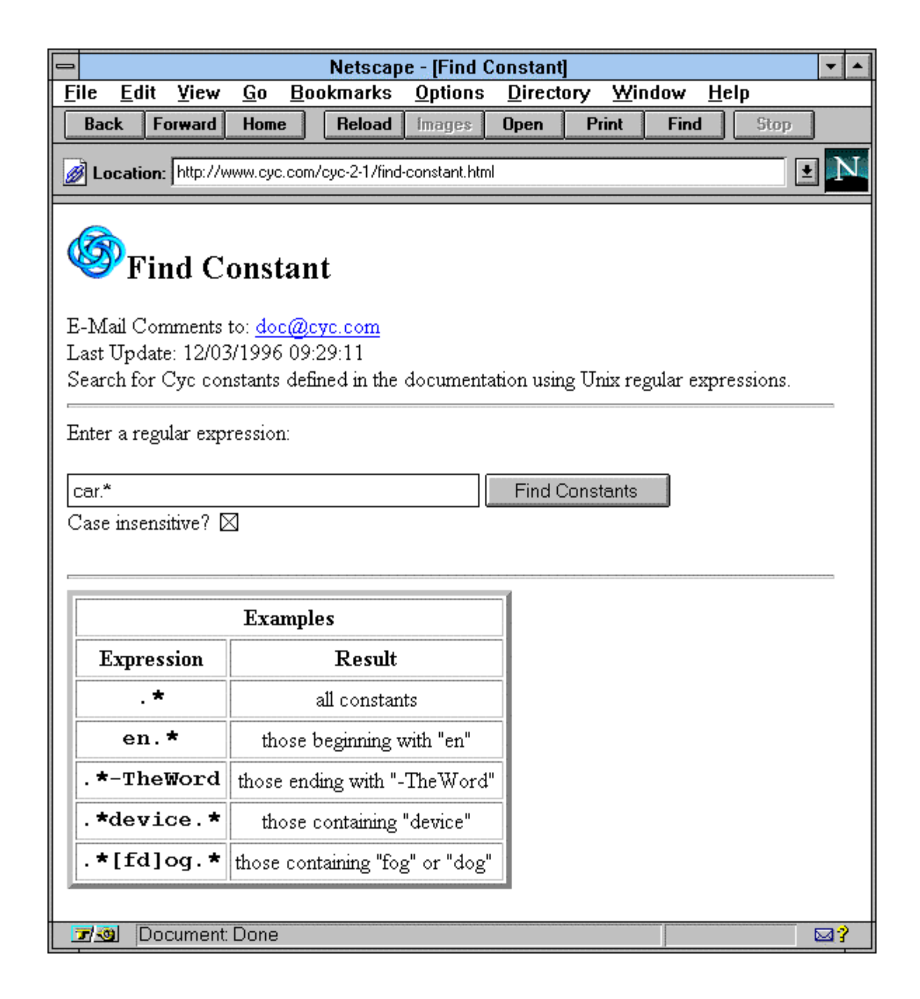


Abbildung 18: Anfrage mit dem regulären Ausdruck car.* (aus Cyc (2001) im März 1997)

3. Welche terminologischen Einträge enthalten den regulären Ausdruck .* ?

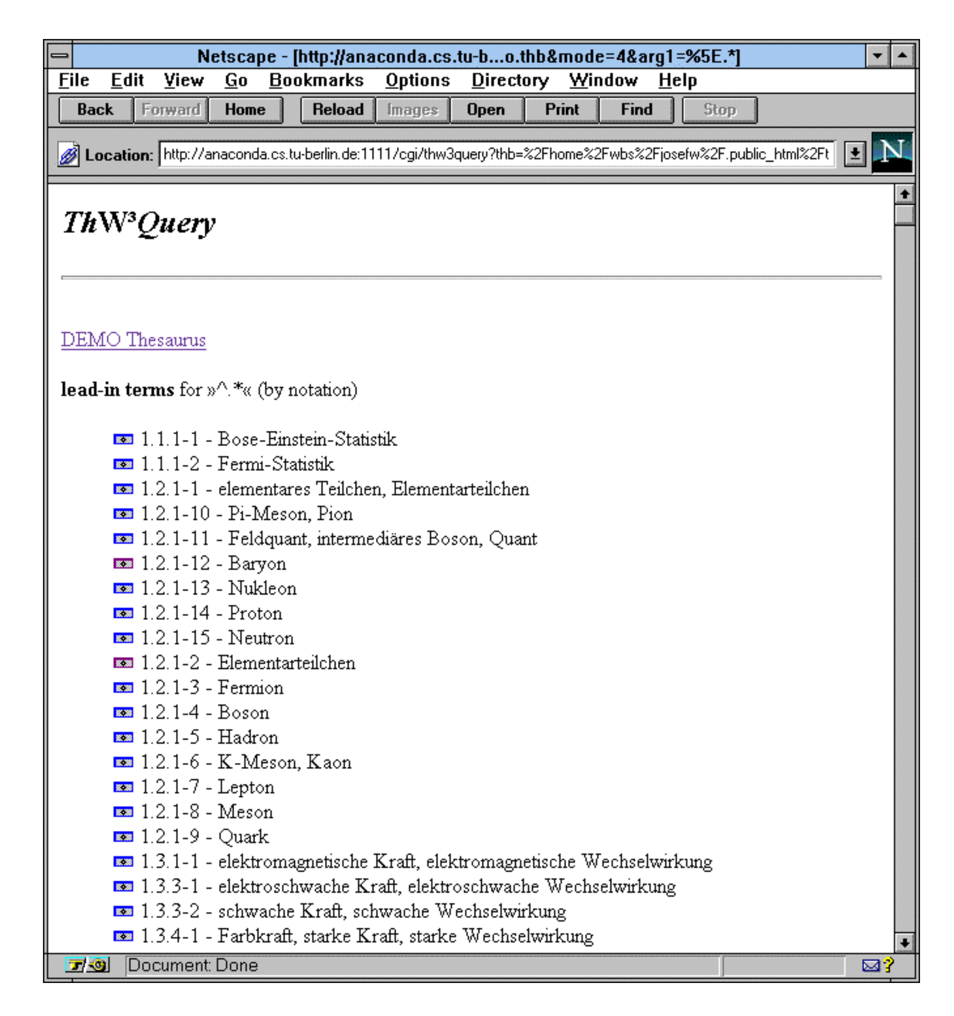


Abbildung 19: Anfrage mit dem regulären Ausdruck .* in THW^{3}Query (aus Hartlep (1996) im März 1997)

4. Welche terminologischen Einträge stehen mit dem terminologischen Eintrag, dessen Name "fahrzeug" ist, in Beziehung?
ISO 8777-Anfrage:
⇒ relate fahrzeug

5. Welche terminologischen Einträge stehen mit dem terminologischen Eintrag, dessen Name "mercedes" ist, in einer Unterbegriffsbeziehung?
ISO 8777-Anfrage:
⇒ relate nt=mercedes

6. Welche terminologischen Einträge vom Typ "Noun" existieren ?

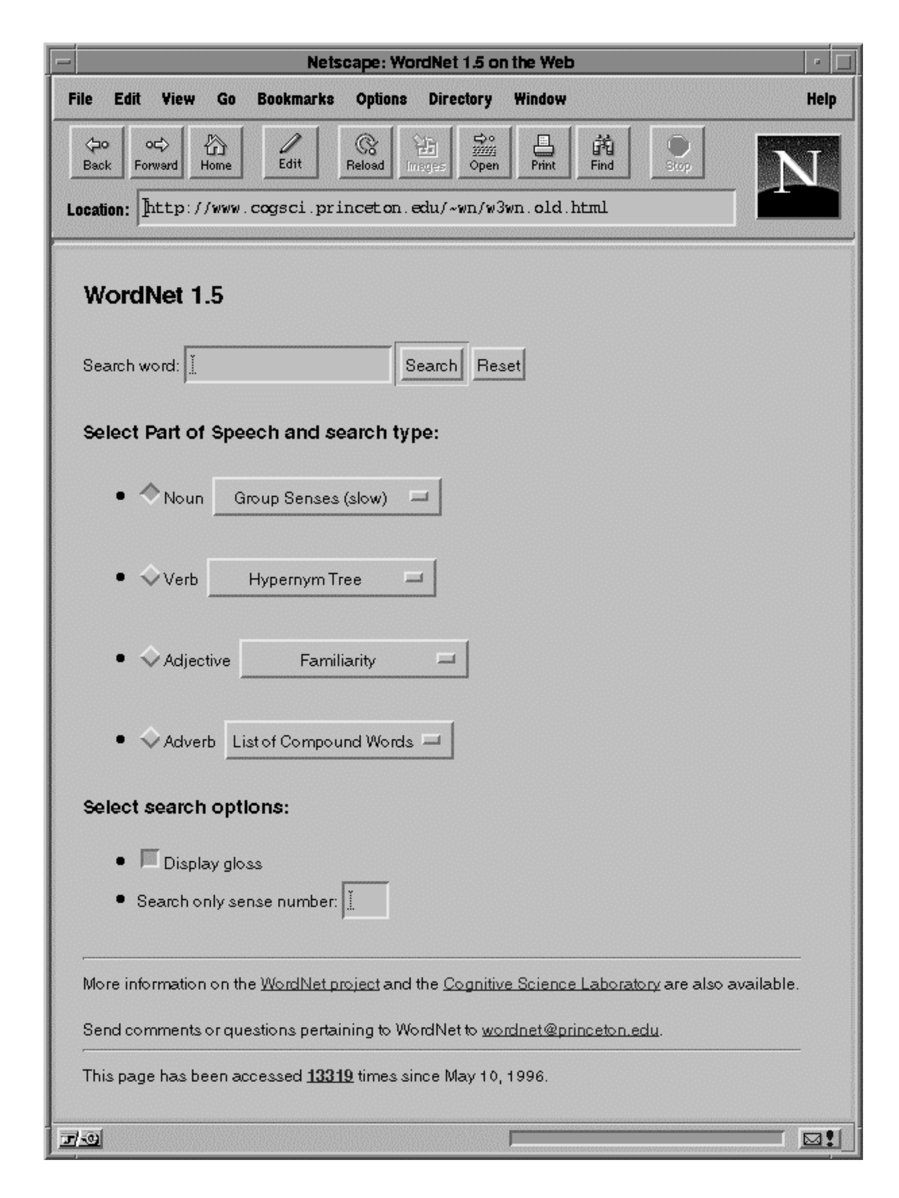


Abbildung 20: Suche nach terminologischen Einträgen (aus Word Net (1997) im März 1997)

Boolesche Suche nach terminologischen Einträgen:

7. Welche terminologischen Einträge enthalten die Zeichenfolge "automobiles" und die Zeichenfolge "vehicles" ?

Netscape - [Thesaurus Search Results]

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Netsite: http://palimpsest.stanford.edu/cgi-bin/search-lctgm?automobiles+and+vehicles

This is a searchable index. Enter search keywords:

Thesaurus Search Results

Index `h-lctgm' contains the following **3** items relevant to `automobiles and vehicles':

- Hearses
- Automobiles
- Vehicles

This page is maintained by Walter Henry.

Document: Done

Abbildung 21: Anfrage mit Thesaurus Graphics (2001) im März 1997

Suche nach Dokumenten

8. Welche Dokumente sind mit dem terminologischen Eintrag, dessen Name "Fahrzeug" ist, indexiert ?

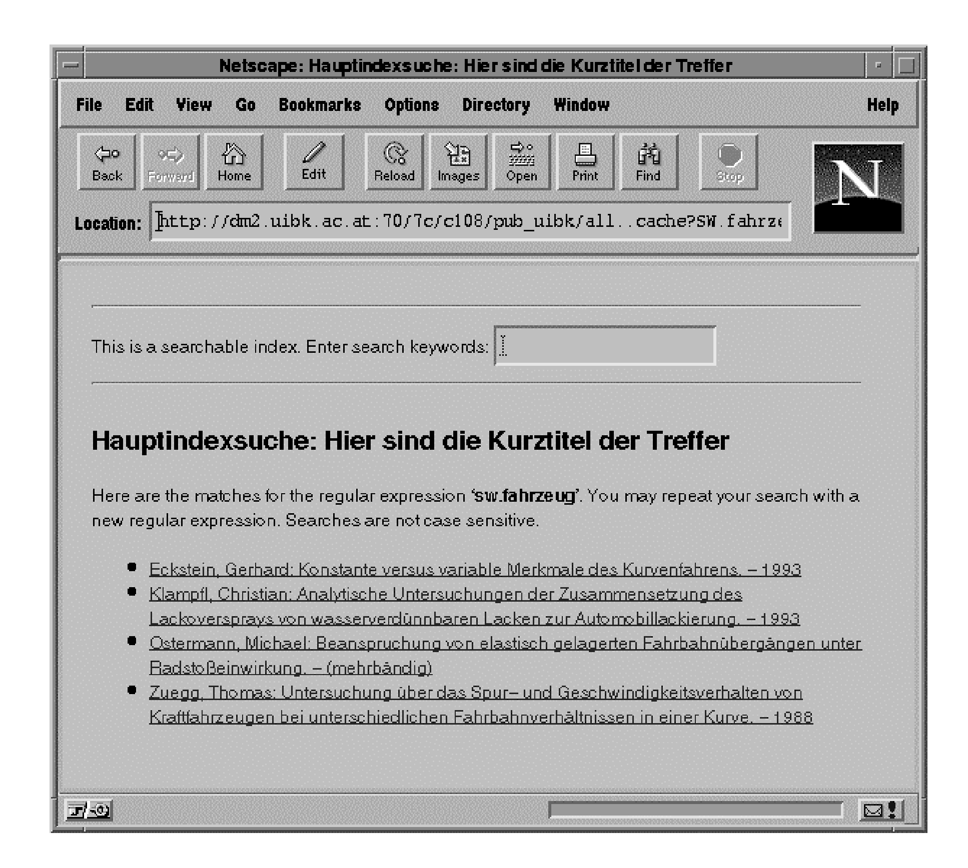


Abbildung 22: Anfrage im Hyperkatalog Innsbruck (im März 1997)

9. Welche Dokumente sind mit einem terminologischen Eintrag, dessen Name "fahrzeug" ist, indexiert oder mit einem seiner Unterbegriffe ?
mit Messenger (2001):
⇒ s fahrzeug+nt/ct

Heuristische Suche
Fischsuche: *"Das Web stellt das Gewässer dar, und in dieses Gewässer werden ein paar Fische an einer Stelle hinzugefügt, von der bekannt ist, daß diese Nahrung enthält. (...) Die Fische vermehren sich nun und die Nachkömmlinge schwimmen in verschiedene Richtungen (...). Diejenigen, die keine neue Nahrung finden sterben aus, das sind also Verweise, die nicht zu neuen relevanten Seiten führen, und diejenigen, die neue Nahrung finden, vermehren sich wieder und suchen weiter nach neuen Nahrungsquellen*." (Bekavac (1996) S. 200)
Die Fischsuche ist also eine beschränkte Tiefensuche, die folgende heuristische Regel benutzt: Beurteile die Relevanz des besuchten Knotens für die Anfrage.
" *(...) die Tendenz, daß die Relevanz der Dokumente abnimmt, je breiter und tiefer der Suchbaum wird."* (Bekavac (1996) S. 200)

3.5.2 Darstellung und Navigation

In der Praxis reichen Anfragen in Form von Kommandos für den Informationssuchenden nicht aus, um relevante Information effizient auffinden zu können. Es sind Verfahren wichtig, die die übersichtliche Darstellung von und die Navigation in terminologischen Wissens- und Dokumentbeständen zum Gegenstand haben. Trotz fehlender oder nicht präsenter Begrifflichkeiten können dadurch Anfragen exakt formuliert und die Suche nach Information, ohne direkt danach gesucht zu haben, Serendipity-Effekt genannt (Kuhlen (1991) S. 38), unterstützt werden.
Zur Verbesserung der Übersichtlichkeit hypermedialer Informationsbestände existieren unterschiedliche Ansätze. Es sind folgende Darstellungsmittel (vgl. Felber, Budin (1989)) für Terminolgien zu unterscheiden:

Alphabetische Darstellung

Benziner
    UB: Pkw
    OB: Fahrzeug
Diesel
    UB: Lkw
    UB: Pkw
    OB: Fahrzeug
    BF: Dieselkraftfahrzeug
Dieselkraftfahrzeug
    BS: Diesel
Elektro
    UB: Lkw
    UB: Pkw
    OB: Fahrzeug
    BF: Elektrofahrzeug
Elektrofahrzeug
    BS: Elektro
Fahrzeug
    UB: Benziner
    UB: Diesel
    UB: Elektro
    UB: Fiat
    UB: Mercedes
    UB: Opel
Familienwagen
    OB: Pkw
Fiat
    UB: Pkw
    OB: Fahrzeug
Lastkraftwagen
    BS: Lkw
Lkw
    OB: Diesel
    OB: Elektro
    OB: Mercedes
    BF: Lastkraftwagen
Mercedes
    UB: Lkw
    UB: Pkw
    OB: Fahrzeug
Opel
    UB: Pkw
    OB: Fahrzeug
Personenkraftwagen
    BS: Pkw
Pkw
    UB: Familienwagen
    UB: Sportwagen
    OB: Benziner
    OB: Diesel
    OB: Elektro
    OB: Fiat
    OB: Mercedes
    OB: Opel
    BF: Personenkraftwagen
Sportwagen
    OB: Pkw

Abbildung 23: alphabetische Darstellung eines Teilthesaurus

Hierarchische Darstellung

- Systematische Darstellung
- Dimensionale Darstellung (Soergel (1974) S. 86)
- Listenförmiger Begriffsplan
- Dezimal-Klassifikation
- Inhaltsverzeichnis
- Bestandsplan
- Fachwerkplan

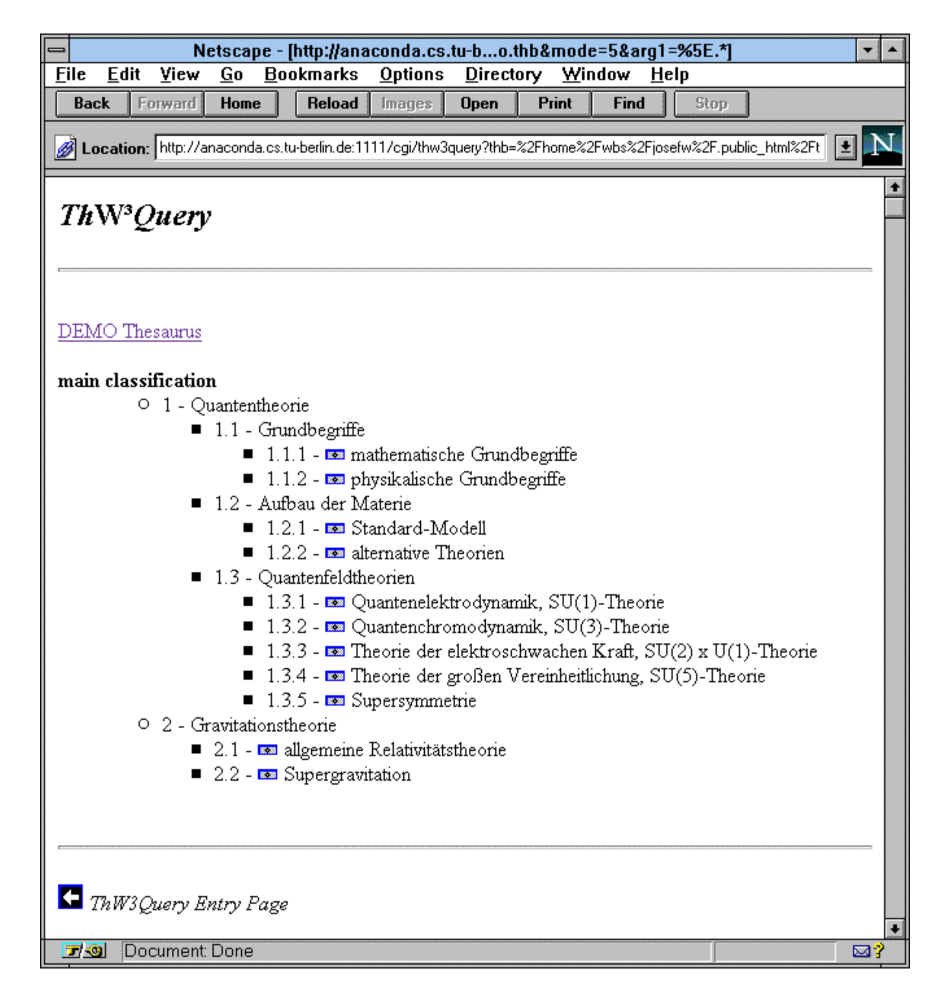


Abbildung 24: Baumdarstellung der Hauptsystematik mit THW³Query (aus Hartlep (1996) im März 1997)

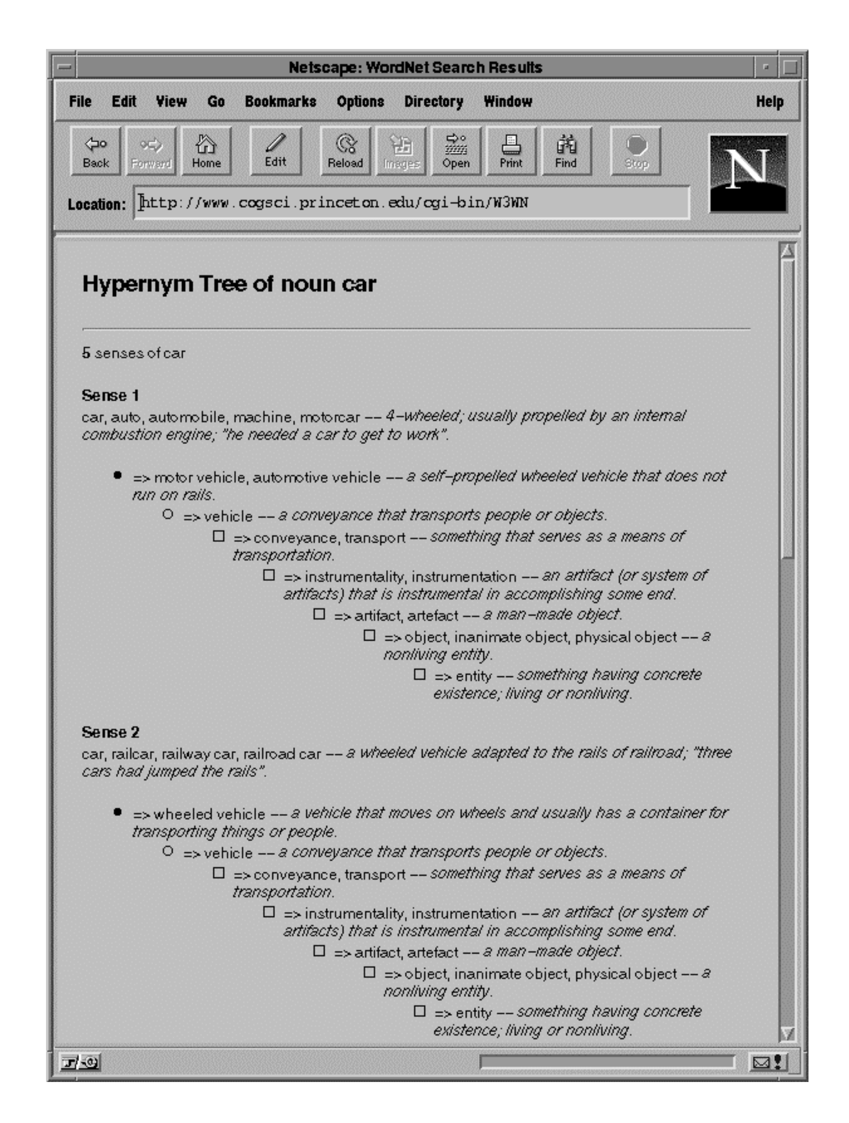


Abbildung 25: Baumdarstellung aller Hyperonyme (ähnlich der Obergriffsrelations) des Substantivs "car" bis hin zur Wurzel (mit einer HTML-Darstellung aus Word Net (1997) im März 1997)

Glossar, Register, Index, Inhaltsverzeichnis, Sachgebietsdarstellung

2. Pflanze, Tier, Mensch (Körperliches)

2.1. Pflanze	2.8. Tier
2.2. Pflanzenarten	2.9. Tierarten
2.3. Pflanzenteile	2.10. Tierzucht
2.4. Pflanzenkrankheiten	2.11. Jagd
2.5. Pflanzenanbau	2.12. Tierkrankheiten
2.6. Fruchtbarkeit	2.13. Mensch
2.7. Unfruchtbarkeit	[...]

Abbildung 26: Sachgebietsdarstellung aus Dornseiff (1959) S. 8

Graphische Darstellung

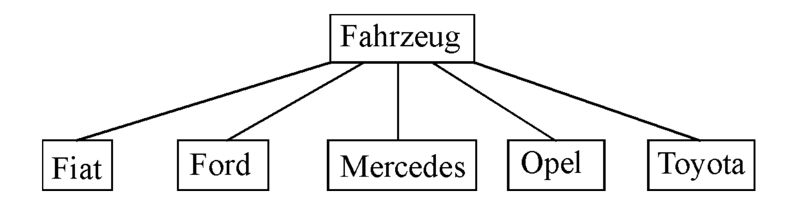


Abbildung 27: Ober/Unterbegriffssystem

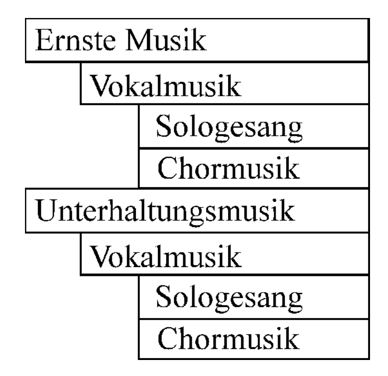


Abbildung 28: Polyhierarchische Listendarstellung

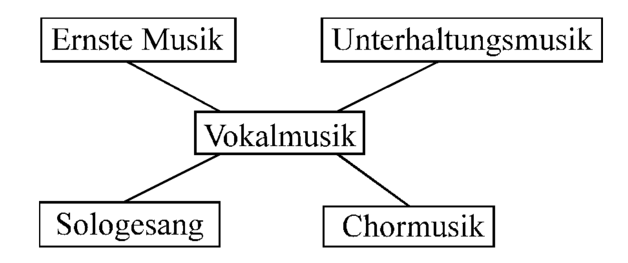


Abbildung 29: Polyhierarchische Graphdarstellung (Form und Inhalt vgl. DIN 1463 S. 8)

Der hier vorliegenden Kreuzklassifikation werden die Modelle jedoch kaum gerecht, da zu kleine Untermengen mit nur einem Merkmal entstehen. Musik kann besser nach Merkmalpaaren wie z.B. ernst-unterhaltend, vokal-instrumental und mit-ohne Solisten eingeteilt werden.

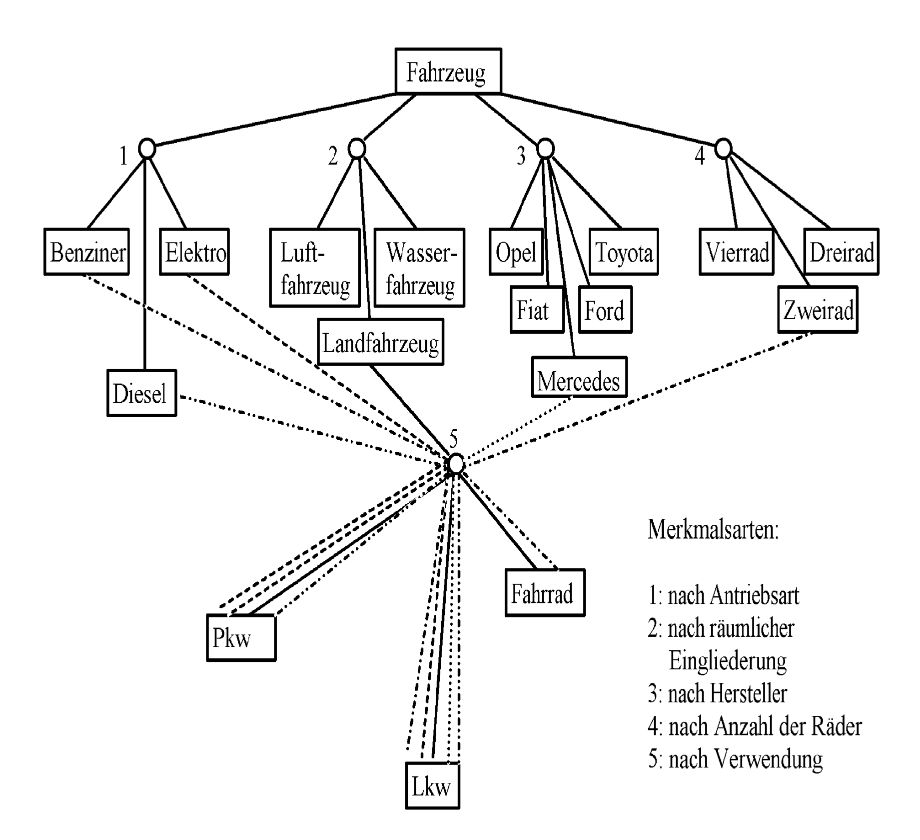


Abbildung 30: Polydimensionale Graphendarstellung (Form vgl. DIN 1463 S. 6)

Diagramm
- Balkendiagramm (gantt chart)
- Baumdiagramm (tree chart)
- Datenflußdiagramm (flow chart)
- Punktdiagramm (dot chart)
- Tabelle (table)
- Tortendiagramm (pie chart)
- Rechteckige und kreisförmige (Pfeil)-Diagramme etc.

Geographische Darstellung
- Karten
- Pläne
- Konstruktionszeichnungen etc.

Darstellung als Text oder Hypertext
Terminologien werden traditionell in einem textuellem Format gespeichert Dabei werden z.T. Textmarkierungen verwendet. Ein weitere Möglichkeit besteht darin, Terminologien in einem Dokumentbeschreibungsformat zu repräsentieren ("Terminology Interchange Format" als SGML-Anwendung nach ISO 12200, ISO 12620, neuerdings auch mit XML). Terminologien können auch mit HTML beschrieben werden. HTML eignet sich jedoch eher zur Darstellung von Dokumenten als zur Repräsentation von Terminologien, da es Metazeichen zur Darstellung wie z.B. Formatierungen enthält. HTML kann beispielsweise dazu eingesetzt werden, Terminologien, die in einem Datenbanksystem gespeichert werden, dynamisch darzustellen. Relationen können mit Hilfe der Markierung "rel" spezifiziert werden. Inverse Relationen

werden mit "rev" (wie reverse) spezifiziert (vgl. Maloney, Quin (1996)). Ein Beispiel ist: Fahrzeug
Maloney, Quin (1996 führen die folgenden Relationen auf: made, home, back, forward, contents, toc (table of contents), index, navigator, child, parent, sibling, top, origin, begin, first, end, last, next, previous biblioentry, bibliography, citation, definition, footnote, glossary, author, copyright, disclaimer, editor, meta, publisher, trademark, banner, bookmark, hotlist, lang, pointer, stylesheet, translation, urc (uniform resource catalogue), node, path.
Im Moment unterstützen die WWW-Oberflächen HTML-Sprachelemente für Relationen nicht adäquat.

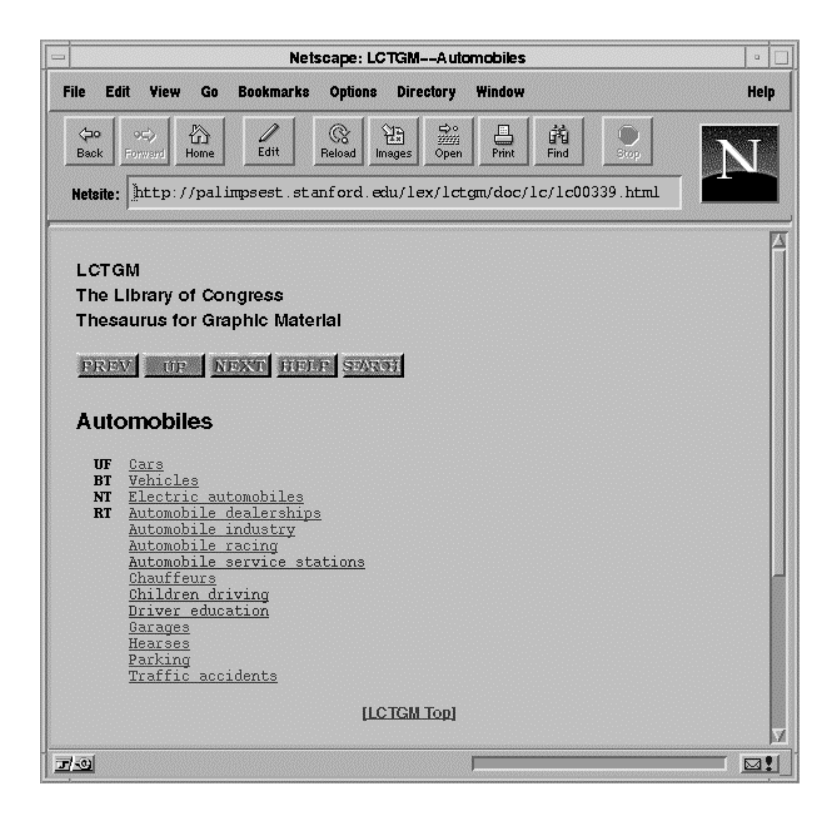


Abbildung 31: Der terminologische Eintrag "Automobiles" und sein terminologischer Kontext (aus Thesaurus Graphics (2001) im März 1997)

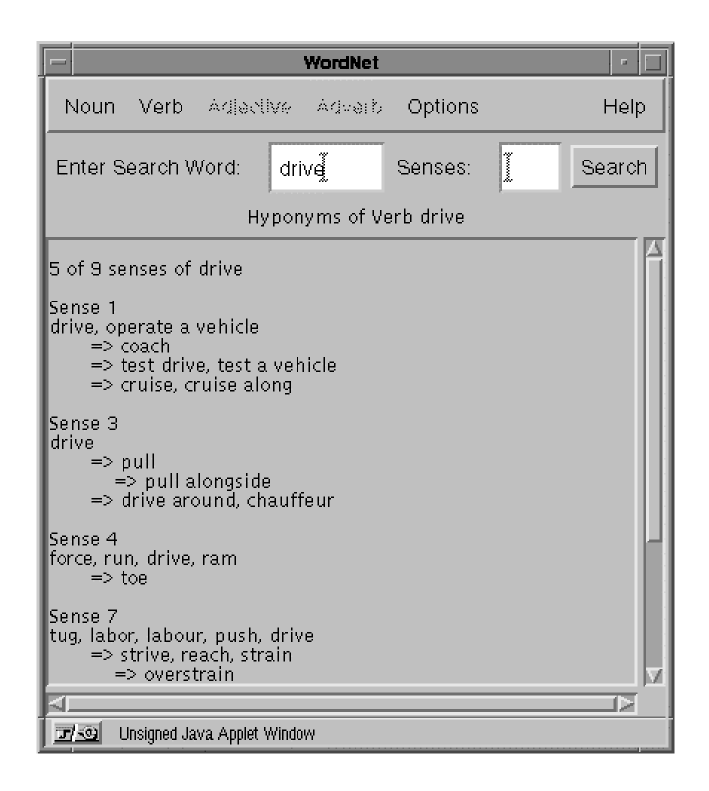


Abbildung 32: Hyponyme (ähnlich der Unterbegriffsrelation) des Verbs "drive" (mit einer Java-basierten Darstellung aus Word Net (1997) im März 1997)

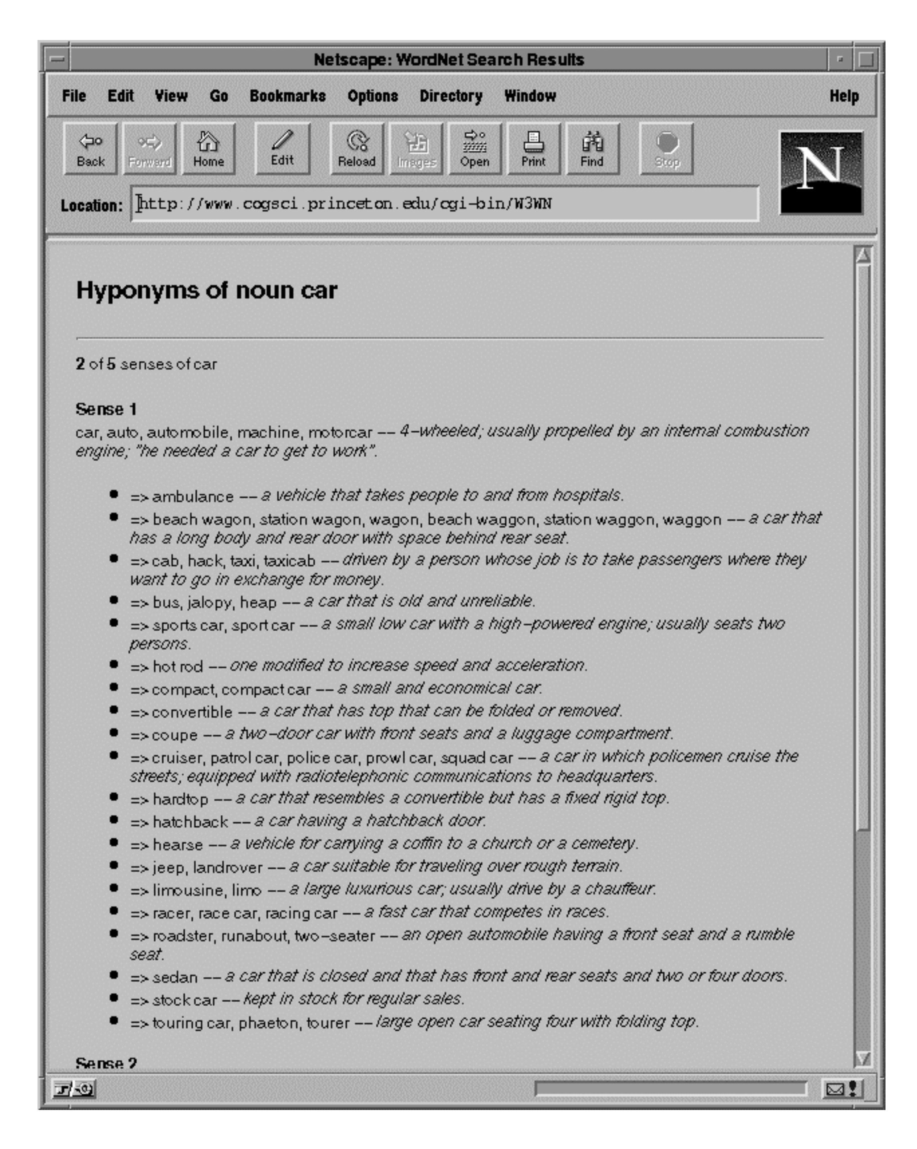


Abbildung 33: Hyponyme (ähnlich der Unterbegriffsrelation) des Substantivs "car" (mit einer HTML-Darstellung aus Word Net (1997) im März 1997)

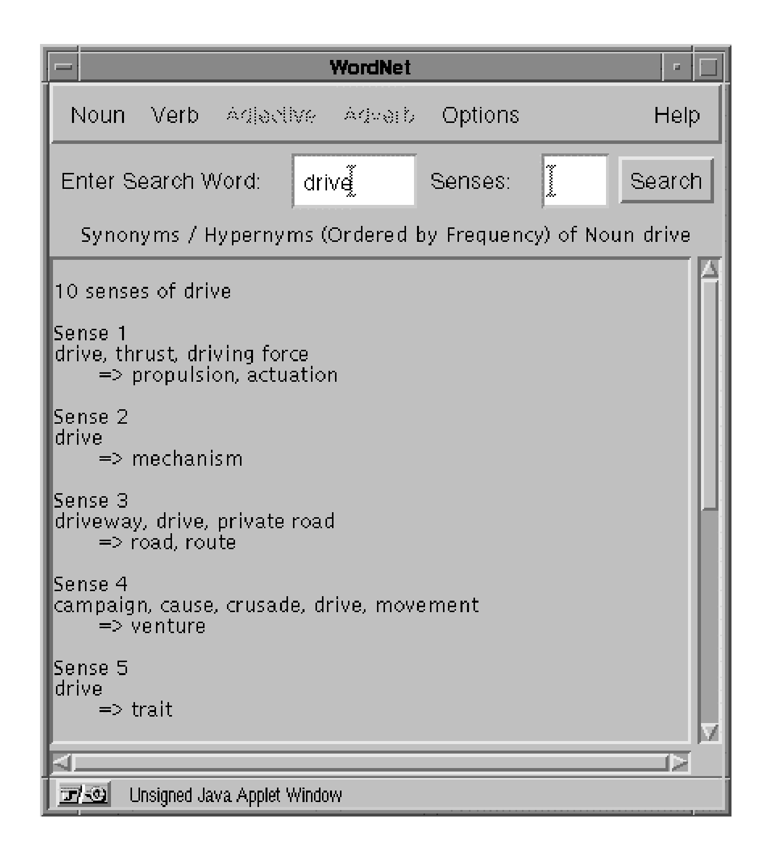


Abbildung 34: Synonyme und Hyperonyme (ähnlich der Oberbegriffsrelation) des Substantivs "drive" (mit einer Java-basierten Darstellung aus Word Net (1997) im März 1997)

Filter

Aus- und Einblenden von Information, Fischaugenprinzip:

- Welche Dokumente wurden von 'Müller' erzeugt
- Welche Nachbartermini existieren von "Europa" bis zur Tiefe 2 entlang der Teil-von-Relation

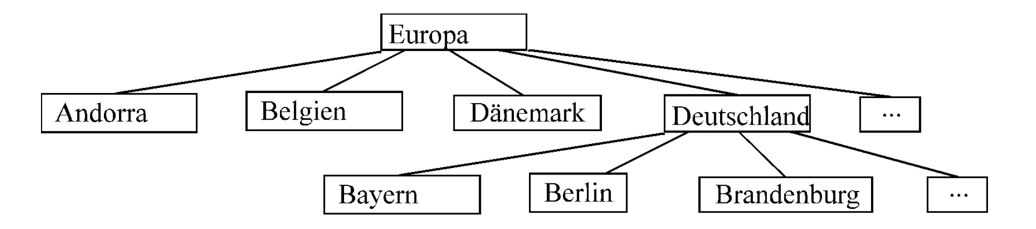


Abbildung 35: Bestandssystem

Markierung

- Schriftart
- Schriftgröße
- Schriftstil
- Farbe etc. beispielsweise werden Relationen fett und Definitionen kursiv angezeigt

Ikonen und Symbole
Durch die Verwendung von Ikonen, Symbolen und anderen multimedialen Elementen kann gegebenenfalls die aufwendige Übersetzung der terminologischen Einträge in die verschiedenen Sprachen vermieden werden.

Sortierung
- Nach alphabetischer Reihenfolge
- Nach Autor
- Nach Entstehungszeitpunkt etc.

Navigationsmittel

Browsing, Wandering
- Gerichtet
- Ungerichtet
- Springen von Terminus-zu-Terminus entlang einer spezifizierten Relation beispielsweise von 'Fahrzeug' zu 'Auto' entlang der Unterbegriffsrelation

Orten
- Die Position in der alphabetischen Anordnung (der 2341'ste Knoten auf S. 20)
- Die Position in der räumlichen Anordnung: die 2. Hierarchiestufe zu 'Fahrzeug' entlang der 'UB'-Relation
- Die Position im Inhaltsverzeichnis (Sachgebiet: Umwelt)

Echolot
- Die Umgebung der momentanen Position abtasten: Welche Nachfahren bis zur Tiefe 2 und vom Typ "Bild" existieren ausgehend von dem terminologischen Eintrag, dessen Name "fahrzeug" ist

History
- Markierung von Knoten und Wegen
- Geführte Wege in ausgesuchten Teilbereichen (guided tour)
- Vom Benutzer beschrittene Wege (breadcrumbs)
- Leser- und autorendefinierte Lesezeichen (Anker werfen)
- Schrittweise Zurücknahme durchgeführter Aktionen
- Den zuletzt besuchten Knoten aufsuchen
- Wohin muß ich noch gehen

Beschreiten paralleler Wege
- von 'Fahrzeug' entlang aller Unterbegriffsrelationen

Ausblenden von Teilbereichen
- die Pfade anzeigen, die den Namen "Unterbegriff" haben

Scanning
- Bereiche überschlagen

Exploring
- einen Sachverhalt vertiefen

Die Kombination von kommandoorientierter Anfrageformulierung mit Darstellungs- und Navigationsmitteln kann das terminologiebasierte Information Retrieval insgesamt verbessern.

3.6 Intelligente Agenten

Unter dem Schlagwort "Intelligente Agenten" sammelt sich eine Vielfalt von Themen. Das Forschungsgebiet ist stark in Bewegung. Wir wollen eine enge Definition eines intelligenten Agenten geben:
Ein intelligenter Agent ist ein Programm, das im Auftrag eines Benutzers Aufgaben ausführt. Ein Agent ist kompetent, kooperativ, kommunikativ, autonom, anpassungsfähig, zielorientiert, mobil, lernfähig, fortlaufend aktiv und besitzt einen eigenen Charakter.
Jennings, Wooldridge (1995) definieren den Informationsagenten:
"An information agent is an agent that has access to at least one, and potentially many information sources, and is able to collate and manipulate information obtained from these sources in order to answer queries posed by users and other information agents (the network of interoperating information sources are often referred to as intelligent and cooperative information systems (...). The information sources may be of many types, including, for example, traditional databases as well as other information agents. Finding a solution to a query might involve an agent accessing information sources over a network." (Jennings, Wooldridge (1995))
Auf dem Gebiet der Informationsagenten sind vor allem folgende Möglichkeiten interessant:
- Standardisierte Kommunikationsprotokolle
- Verteilte Kommunikation, parallele Anfragen
- Nutzung heterogener Informationsbestände
- Operatoren zur Navigation in Informationsbeständen
- Wissensbasierte Suchmöglichkeiten, Berücksichtigung von Benutzerwissen
- Zeitversetzte Anfragekommunikation, Relevanzfeedback, Frageverfeinerung
- Präsentation der Suchergebnisse als Hypertext, Zusammenführung von Suchergebnissen mehrerer Informationsagenten etc.

Kommunikation zwischen Benutzern und Agenten bzw. zwischen Agenten kann mit Hilfe der Sprechakttheorie beschrieben werden.
In diesem Zusammenhang wurde die Sprache KQML (knowledge query and manipulation language; Beck et.al. (1993)) geschaffen. KQML formalisiert z.B. Fragedialoge wie: jemandem eine Ja-Nein-Frage stellen, jemandem eine Welche-Frage stellen, auf eine Frage von jemandem antworten, auf eine Frage von jemandem mit sorry antworten, jemanden um Frageverfeinerung bitten etc. Basis für KQML ist die logische Sprache KIF (knowledge interchange format; Finin, Genesereth (1992)). KIF ist eine Erweiterung des Prädikatenkalküls erster Stufe.
Informationsagenten sind beispielsweise Autonomy (2001), InfoMagnet (2001), InfoSleuth (Jacobs, Shea (1996)), WebWhacker (2001) etc. Weitere werden in UMBC (2001) aufgelistet.

Beispielanfragen

1. Welche Dokumente bis zur Suchtiefe "4", ausgehend vom Dokument mit der URL "http://www.cs.tu-berlin.de/", existieren und enthalten nicht die Zeichenfolge "wbs" in ihrer URL ?
Nach Herstellerangaben ist die Anfrage (bis auf den negierten Teil der Anfrage) mit WebWhacker (1997) möglich.

2. Welche Dokumente interessieren mich (mein Benutzerprofil) zum Thema "Verbrechen" (z.B. die Dokumente der Orte, auf die ich besonders häufig zugreife oder die ich gesondert markiere) ?
Nach Herstellerangaben ist die Anfrage mit AutonomyAgentWare (1997) und InfoMagent (1997) möglich.

3. Welche Dokumente handeln von "Superusereingriffen" und sind von deutschen Suchmaschinen indexiert ? Die Anfrage soll ab heute periodisch alle zwei Wochen gestellt und das Ergebnis lokal gespeichert werden. Im Suchergebnis soll das Fehlen von Dokumenten im Vergleich zu den vorherigen Suchergebnissen markiert werden.
Nach Herstellerangaben ist die Anfrage (bis auf die Markierung fehlender Dokumente) mit WebWhacker (1997) und Netriever (1997) möglich.

3.7 Vergleich der untersuchten Systeme

Wir vergleichen die untersuchten Systeme, indem wir deren Feld/Formatstruktur, Bestand, Suchmittel, Suchergebnisse und Leistungsgrenzen zusammenfassen.

<table>
<tr><th colspan="2">Dokumenttyp</th><th>Feld/Formatstruktur</th></tr>
<tr><td rowspan="7">einfach strukturiertes Dokument</td><td>Person/Gruppe</td><td>Felder von natürlichen, virtuellen und juristischen Personen, Felder von Firmen, Institutionen, Organisationen, Vereinen, juristischen Einzelpersonen,und virtuellen Gruppen (siehe Kap. 3.1.1);
Formate: RFC 742, RFC 812, RFC 954</td></tr>
<tr><td>Rechner</td><td>Felder von Rechnern und Netzwerken (siehe Kap. 3.1.2);
Formate: RFC 1034</td></tr>
<tr><td>Datei</td><td>Felder von Dateien (siehe Kap. 3.1.3)
Formate: c, doc, dvi, eps, exe, frame-maker, html, man, mbox, me, message, ms-word, pl, ps, rtf, sgml, src, tex, txt, word-perfect, xml, ...</td></tr>
<tr><td>E-Mail</td><td>Felder von E-Mails (siehe Kap. 3.1.4);
Formate: RFC 821, ISO 10021</td></tr>
<tr><td>Artikel</td><td>Felder von Artikeln und Artikelgruppen (siehe Kap. 3.1.5);
Formate: RFC 977</td></tr>
<tr><td>Menü</td><td>Felder von Menüs (siehe Kap. 3.1.6);
Formate: RFC 1436</td></tr>
<tr><td>Dokument-Hinweis</td><td>Felder von Dokumenten in Bibliothekskatalogen (siehe Kap. 3.1.7);
Formate: ISO 8777, UNIMARC (1997), USMARC, ISO 2709, MAB (1997), Dublin Core (Daniel et.al. (1995))</td></tr>
<tr><td>link-strukturiertes Dokument</td><td colspan="2">Felder von link-strukturierten Dokumenten, Netzwerkstruktur der Hypertext-Dokumente
Formate: html, htf, xml, sgml</td></tr>
<tr><td>geschachteltes Dokument</td><td colspan="2">Felder geschachtelter Dokumente (auch multimediale Dokumente), Dokument Typ Definitionen (DTD), siehe auch Kap. 3.3 und Kap. 3.1.8
Formate:
- Bild: g3fax, gif, jpeg, png, pbm, pgm, ppm, rasterfile, rgb, tiff, xbitmap, xpm, xwd, ...
- Ton: aiff, au, midi, mpeg, pn-realaudio, raw, snd, wav, ...
- Film: avi, fli, mpeg, quicktime, ...
- 3-D-Objekt: vrml, ...
- geschachtelte Dokumente: html, xml, sgml</td></tr>
<tr><td>Fakten</td><td colspan="2">Beschreibung der Relationen (siehe Kap. 3.4)</td></tr>
<tr><td>Terminologische Einträge</td><td colspan="2">Felder von Einträgen, Netzwerkstruktur (siehe Kap. 3.5)</td></tr>
</table>

Tabelle 5: Feld/Formatstruktur von Internet-Dokumenten

Suchart		Informationssysteme
Suche in einfach struktu-rierten Doku-menten	Person/Gruppe	finger, whois, netfind, X.500, ...
	Rechner	DNS, whois, netfind, ...
	Datei	Dateisysteme, Telnet, FTP, Alex, Archie, ...
	E-Mail	mail, elm, Netscape-Messenger, MS-Outlook, ...
	Artikel	Newssysteme, Suchmaschinen, ...
	Menü	Gopher, ...
	Dokument-Hinweise	OCLC, Horizon, Aleph, ...
	Volltextsuche	WAIS, Harvest, Suchmaschinen, ...
Suche in link-strukturierten Dokumenten		HTTP/URL, Hyper-G, W3QS, WebSQL, ...
Suche in geschachtelten Dokumenten		TSIMMIS, XQL, Intermedia, ...
Faktensuche		Datenbanksysteme: Oracle, DB2, Informix, ... Fakteninformationssysteme: STN, DIMDI, NlightN, ...
Suche nach Terminologi-schen Einträgen		Thesaurussysteme, Terminologie- und Wörterbuchsysteme, ...
Intelligente Agenten		Expertensysteme, Agentensysteme, ...

Tabelle 6: Internet-Informationsysteme

Suchart		Bestand
Suche in einfach struktu-rierten Doku-menten	Person/Gruppe	Rechner, Organisation, Region, Land, global
	Rechner	Netzwerk, Organisation, Land, global
	Datei	Datenträger, logisches Dateisystem, globale FTP-Bestände
	E-Mail	Persönliches Archiv
	Artikel	Artikelgruppe, Teilmenge von Artikelgruppen, global, Zeit-raum
	Menü	Gopher-Server, global
	Dokument-Hinweise	Katalog einer Bibliothek, Katalog mehrerer Bibliotheken, Katalog eines Bibliotheksverbunds, global
	Volltextsuche	Datenbank, mehrere Datenbanken, Server, benannte Kollek-tionen, Region, Land, Sprache, global, Zeitraum
Suche in link-strukturierten Dokumenten		benannte Kollektionen, Server, Region, Land, global, von einem Dokument navigierbare Dokumente (einer best. Such-tiefe), Zeitraum
Suche in geschachtelten Dokumenten		einzelnes XML-Dokument, Datenbank, mehrere Datenbanken
Faktensuche		Datenbank-Benutzer, Datenbank, Server
Suche nach Terminologi-schen Einträgen		Fachgebietssammlung, Universalsammlung, spezifische Sammlungen
Intelligente Agenten		siehe Suche in link-strukturierten Dokumenten, siehe Suche in geschachtelten Dokumenten, siehe Faktensuche

Tabelle 7: Bestand von Internet-Informationssystemen

Suchart			Suchmittel[7]
Suche in einfach struktu-rierten Doku-menten	Person/Gruppe		einfache Volltextsuche mit Attributen
	Rechner		einfache und reguläre Volltextsuche mit Attributen
	Datei		einfache Volltextsuche mit Attributen; reguläre Volltextsu-che für das Attribut Dateiname; Nachfahren einer Datei, logische Und-Verknüpfung teilweise möglich
	E-Mail		einfache Volltextsuche mit Attributen; logische Und-Verknüpfung teilweise möglich
	Artikel		einfache Volltextsuche mit Attributen; logische Und- bzw. -Oder-Verknüpfungen
	Menü		einfache Volltextsuche mit Attributen; komplexe boolesche Verknüpfungen
	Dokument-Hinweise		einfache Volltextsuche mit Attributen (zusätzlich: Links, und Innenmaskierung, Unterscheidung von Groß/Kleinschreibung); komplexe boolesche Verknüpfun-gen; Wortabstandsoperatoren: near, before und adjacent (maximale Wortabstandszahl spezifizierbar)
	Voll-text-suche	WAIS	einfache Volltextsuche mit Attributen; phonetische Ähnlich-keit von Zeichenfolgen; Wortabstandsoperator, Worthäufig-keiten
		Harvest	reguläre Volltextsuche mit Attributen; komplexe boolesche Verknüpfungen
		Suchma-schinen	einfache Volltextsuche mit Attributen; Unterscheidung von Groß/Kleinschreibung; phonetische Ähnlichkeit, Namenser-kennung, Synonyme; Wortabstandsoperationen; Anhäu-fungs-, Plus- und Minus-Operator; Operator für Vorfahren einer Stufe; komplexe boolesche Operatoren
Suche in link-struktu-rierten Doku-menten	HTTP/URL		Suche mit dem Attribut URL
	Hyper-G		Suche mit dem Attribut URL; einfache Volltextsuche mit Attributen, Volltextsuche über alle Nachfahren; boolesche Verknüpfungen
	W3QS		SQL-ähnlich; reguläre Volltextsuche mit Attributen (mit der Programmiersprache PERL); Nachfahren von einem Doku-ment ausgehend; Suchtiefe und Gesamtzahl zu untersuchen-der Knoten angebbar; boolesche Verknüpfungen: and, or
	WebSQL		SQL-ähnlich; reguläre Volltextsuche mit Attributen; boole-sche Verknüpfungen; Pfadoperatoren (lokale/globale Ver-weise, Suchtiefe etc.)

Tabelle 8a: Suchmittel der Internet-Informationssysteme

[7] Eine einfache Volltextsuche ist eine Wortsuche mit Rechtsmaskierung, eine reguläre Volltextsuche ist eine Volltextsuche mit regulären Ausdrücken (siehe Kap.2.1.4).
Mit boolesche Verknüpfungen sind die zweistelligen Operatoren and, or, und andNot gemeint.
Mit komplexen booleschen Verknüpfungen ist die Kombination von booleschen Ausdrücken gemeint.

Suche in geschach-telten Do-kumenten	TSIMMIS	SQL-ähnlich; boolesche Verknüpfungen; Pfadoperatoren für geschachtelte Objekte (. , ? , *)
	XQL	Operatoren für Teile (XML-Elemente) eines XML-Dokuments: Teile der Stufe n ("/"), alle Teile ("//"); Operator für die Stelle im XML-Dokument ("[n]"); Vergleichsoperatoren für Werte von XML-Elementen oder XML-Attributen: Gleichheit, Ungleichheit, kleiner als, größer als, usw.; boolesche Verknüpfungen; logisches einstelliges nicht; Quantoren: es existiert ein, für alle; Mengenoperatoren: Vereinigung, Schnittmenge
	Intermedia	einfache Volltextsuche mit Attributen; zusätzlich Links- und Innenmaskierung; Unterscheidung von Groß/Kleinschreibung; Wortabstandsoperatoren; phonetische Ähnlichkeit, Rechtschreibähnlichkeit, Vollformenerweiterung; Anhäufungs- und Minus-Operator; komplexe boolesche Verknüpfungen; komplexe terminologische Operatoren; thematische Anfragen
Faktensuche		SQL, Messenger, Grips, objektorientierte Anfragen etc.
Suche nach Terminologischen Einträgen		einfache Volltextsuche nach Einträgen, komplexe Darstellungs- und Navigationsmöglichkeiten (siehe Kap. **3.5**)
Intelligente Agenten		einfache Volltextsuche mit Attributen; Volltextsuche mit regulären Ausdrücken; Ausgangspunkt (URL), Suchtiefe, Suchbreite; periodische Anfragen (Zeitpunkt, Wiederholungshäufigkeit); Aufteilung von Anfragen an Suchmaschinen und weitere Agenten (interne und externe Retrievalagenten); Berücksichtigung von Benutzerwissen (Benutzerprofil, Relevanzfeedback), terminologisches Wissen; automatische Klassifizierung; Navigationsmittel (go, go to, show, jump to, ... , left, right, north, south, ..., floor, building) etc.

Tabelle 8b (Forts.): Suchmittel der Internet-Informationssysteme

Suchart		Anzeige	Sortierung
Suche in einfach strukturierten Dokumenten	Person/ Gruppe	Teilmenge der erfassten Felder	nach dem Namen
	Rechner	Teilmenge der erfassten Felder	nach dem Namen
	Datei	Teilmenge der erfassten Felder, Textzeilen, Anzahl von Dateien	nach mehreren Feldern
	E-Mail	Teilmenge der erfassten Felder	nach mehreren Feldern
	Artikel	Teilmenge der erfassten Felder	nach Datum, Subject und weiteren Feldern
	Menü	Teilmenge der erfassten Felder	nach dem Namen
	Dokument-Hinweis	Teilmenge der erfassten Felder	nach mehreren Feldern
	Volltextsuche	Teilmenge der erfassten Felder mit Positionsangaben; Textzeilen; Anzahl von Dateien	nach mehreren Feldern; nach unterschiedlichen Ähnlichkeitsfunktionen
Suche in linkstrukturierten Dokumenten		Teilmenge der erfassten Felder	nach mehreren Feldern; nach unterschiedlichen Ähnlichkeitsfunktionen
Suche in geschachtelten Dokumenten		Teilmenge der erfassten Felder mit Positionsangaben; Textzeilen; Anzahl von Dateien	nach mehreren Feldern; nach unterschiedlichen Ähnlichkeitsfunktionen
Faktensuche		siehe SQL	siehe SQL
Suche nach terminologischen Einträgen		Teilmenge der Felder; Nachbareinträge, hierarchische und graphische Darstellungen	nach mehreren Feldern, hierarchisch etc.
Intelligente Agenten		Teilmenge der erfassten Felder; Netzwerkstruktur der Hypertext-Dokumente (guided tour, zusammenhängend, optimiert); maximale Größe (Anzahl der Dokumente, Suchtiefe und -breite) des Ergebnisses spezifizierbar; Zusammenführung der Suchergebnisse von Informationsagenten; Suchergebnis kann an weitere Personen/Gruppen gesendet werden; periodische Suchergebnisse etc.	nach mehreren Feldern; nach unterschiedlichen Ähnlichkeitsfunktionen; Kennzeichnung relevanter Dokumente;

Tabelle 9: Suchergebnisse in Internet-Informationssystemen

Suchart			Leistungsgrenzen
Suche in einfach strukturierten Dokumenten	Person/Gruppe		meist keine Unterscheidung von Groß/Kleinschreibung; vorwiegend 7 Bit-Zeichensatz; keine boolesche Verknüpfung der Attribute möglich
	Rechner		meist keine Unterscheidung von Groß/Kleinschreibung; vorwiegend 7 Bit-Zeichensatz; meist keine Spezifizierung von Attributen bei der Suche möglich (Suche über alle Attribute)
	Datei		siehe Kap. 3.1.3
	E-Mails		meist keine Unterscheidung von Groß/Kleinschreibung; vorwiegend 7 Bit-Zeichensatz; Keine Spezifizierung von Mime-Attributen möglich
	Artikel		meist keine Unterscheidung von Groß/Kleinschreibung; Vorwiegend 7-Bit-Zeichensatz; keine regulären Volltextanfragen möglich; meist keine Spezifizierung von Attributen bei der Suche möglich (Suche über alle Attribute); keine einheitlichen booleschen Verknüpfungen möglich (eingeschränkte boolesche Verknüpfungen sind mit Dejanews möglich, mit Netnews und Tin können boolesche Verknüpfungen zum Teil durch Anfragen in einer bestimmten Reihenfolge nachgebildet werden); keine komplexen booleschen Verknüpfungen; keine Operatoren für Nachfahren, Vorfahren, Verweise etc.
	Menü		keine Unterscheidung von Groß/Kleinschreibung; 8-Bit-Zeichensatz; keine regulären Volltextanfragen auf Attributen möglich; kein einstelliges logisches not möglich; keine Volltextanfragen für den Gesamtinhalt möglich; keine Operatoren für Nachfahren, Vorfahren, Verweise etc.
	Dokument-Hinweis		8-Bit-Zeichensatz; keine regulären Volltextanfragen auf Attributen möglich; kein einstelliger boolescher Operator "not", keine Volltextanfragen für den Gesamtinhalt möglich; kein Operator für Nachfahren, Vorfahren, Verweise etc.
	Volltextsuche	WAIS	8-Bit-Zeichensatz; keine Volltextanfragen mit regulären Ausdrücken; kein einstelliger boolescher Operator not; keine Operatoren für Nachfahren, Vorfahren, Verweise etc.
		Harvest	8-Bit-Zeichensatz; keine Anfragen mit phonetischer Ähnlichkeit, mit Wortabständen und mit Worthäufigkeiten; kein boolescher Operator "not" möglich (weder einstellig noch zweistellig)
		Suchmaschinen	8-Bit-Zeichensatz; keine Anfragen mit phonetischer Ähnlichkeit und mit Worthäufigkeiten; z.T. keine Verwendung von Stopwörtern möglich

Tabelle 10a: Leistungsgrenzen von Internet-Informationssystemen

Suche in link-struktu-rierten Doku-menten	HTTP/URL	8-Bit-Zeichensatz; beschränkt auf die eindeutige Referenzierung von Dokumenten
	Hyper-G	8-Bit-Zeichensatz; keine Unterscheidung von Groß/Kleinschreibung; keine regulären Volltextanfragen möglich; Stopwörter können in der Volltextsuche nicht verwendet werden; Zeichenkombinationen ("ae", "oe", ...) werden bei der Volltextsuche auf die Zeichen ("ä", "ö", ...) abgebildet; keine einheitlichen booleschen Anfragen; boolesche Anfragen und Suche mit Attributen nur im Suchraum des Hyper-G-Bestandes; kein einstelliger boolescher Operator "not" möglich; Operator für Nachfahren kann nicht universell für alle Anfragen verwendet werden; kein Operator für Vorfahren, Verweise etc.
	W3QS	8 Bit Zeichensatz; kein zweistellige Verknüpfung andNot und kein einstelliger boolescher Operator not; kein Operator für Vorfahren, Verweise (einer best. Beziehungsart) etc.
	WebSQL	8 Bit-Zeichensatz (Erweiterung auf Unicode durch Java einfach möglich); kein einstelliger boolescher Operator not; kein Operator für Vorfahren
Suche in ge-schacht elten Doku-menten	TSIMMIS	8 Bit Zeichensatz; keine Volltextanfragen; kein einstelliger boolescher Operator not möglich; kein Operator für Verweise (Nachfahren, Vorfahren etc.)
	XQL	8 Bit Zeichensatz; Beschränkung auf ein Dokument; kein Operator für Verweise (Nachfahren, Vorfahren etc.); keine komplexen Volltextanfragen
	Intermedia	Unicode-Zeichensatz; kein Plus-Operator; kein einstelliger boolescher Operator not; kein Operatoren für Verweise (Nachfahren, Vorfahren etc.), keine regulären Volltextanfragen
Faktensuche		Unicode-Zeichensatz; keine regulären Volltextanfragen; keine Operatoren für Nachfahren, Vorfahren, Verweise etc.
Suche nach terminologischen Einträgen		aufgrund der Diversifizierung der Systeme keine Angabe möglich
Intelligente Agenten		aufgrund der Diversifizierung der Systeme keine Angabe möglich

Tabelle 10b (Forts): Leistungsgrenzen von Internet-Informationssystemen

4 Aufbau von Anfragesprachen für Internet-Informationssysteme

Für den Aufbau der Sprachen wird empirisches Material aus der Untersuchung von Internet-Informationssystemen verwendet (siehe Kap. 3). Ein großer Teil der dort kristallisierten Anfragemöglichkeiten wird in den Anfragesprachen abgebildet. Einige Beispielanfragen aus der Untersuchung werden übernommen. Aus Komplexitätsgründen werden die Systemklassen Faktensuche und Intelligente Agenten nicht in die Arbeit einbezogen.
Die aufgebauten Anfragesprachen sind nicht für den Benutzer, sondern als Zwischensprachen zwischen Benutzeranfragesprachen und den Informationssystemen konzipiert. Nur ein Rechercheexperte benutzt die Anfragesprachen direkt.
Im ersten Kapitel wird eine Anfragesprache zur Suche in einfach strukturierten Dokumenten (SimpleStructuredQL) aufgebaut, im zweiten und dritten Kapitel werden Anfragesprachen zur Suche in link-strukturierten Dokumenten(SimpleLinkQL, LinkQL) aufgebaut und im vierten Kapitel wird eine Anfragesprache zur Suche in geschachtelten und link-strukturierten Dokumenten (Nested&LinkQL) aufgebaut. Im fünften Kapitel werden die Anfragesprachen zu einer universellen Anfragesprache zur Suche in strukturierten Dokumenten (StructuredQL) zusammengeführt.
Für jede Anfragesprache wird deren Syntax und Semantik exakt spezifiziert. Anhand von Beispielen wird vorgeführt, welche Art von Anfragen mit der Sprache gestellt werden können und in welchen Schritten das Ergebnis der Anfrage (Semantik) berechnet wird.
Die aufgebauten Anfragesprachen sind prädikatenlogische Anfragesprachen. Anfragen werden mit dem Lambda-Operator λ und Formeln gebildet. In den Anfragesprachen werden 2-stellige Prädikate verwendet. Grundsätzlich könnten auch n-stellige Prädikate verwendet werden. Diese lassen sich jedoch durch 2-stellige Prädikate nachbilden (Nilsson (1982) S. 363-369).
Konventionen zur Schreibweise:
Syntax: Sprachelemente werden klein geschrieben (Individuenkonstanten und -variablen, Funktionskonstanten- und variablen). Bei den Prädikatenkonstanten wird der erste Buchstabe groß und der darauffolgende Teil klein geschrieben. Prädikatenvariablen werden ausschließlich mit dem Großbuchstaben P bezeichnet. Nummerierungen werden hinter das Sprachelement tiefgestellt. Sorten werden klein geschrieben und hinter die Individuen-, Funktions- und Prädikatenkonstanten und -variablen hochgestellt. Da sich die Stelligkeit eines Prädikats implizit durch die Anzahl der Argumente ergibt, werden diese nicht zusätzlich explizit angezeigt.
Semantik: Individuenbereiche und deren Elemente werden groß geschrieben. Ausnahme sind Zeichenfolgen. Diese werden in Hochkommata eingeschlossen. Der Individuenbereich *U* wird kursiv geschrieben damit er von Elementen von *U* (z.B. U_1) unterschieden werden kann. Nummerierungen werden hinter das Element tiefgestellt.

4.1 SimpleStructuredQL: Eine Anfragesprache für einfach strukturierte Dokumente

Im Internet werden unterschiedliche einfach strukturierte Dokumente bereitgestellt: E-Mails, Personen/Gruppen, Rechner, Dateien, Artikel, Menüs, Dokumenthinweise etc. Für diese Bestände wird die Anfragesprache SimpleStructuredQL aufgebaut. Anwender von SimpleStructuredQL sind Benutzer, die einen einheitlichen Zugriff auf einfach strukturierte Internet-Dokumente wünschen. Für die Anfragesprache werden Ergebnisse aus Reiner (1991) S. 95-106 übernommen.

SimpleStructuredQL enthält Sprachelemente für Deskriptoren und Vergleichsoperatoren (Zahlenvergleiche, Volltextoperatoren, linguistische Operatoren und Thesaurusoperatoren). Die Namen der Deskriptoren werden zum größten Teil aus der Untersuchung der Informationssysteme (siehe Kap. 3) übernommen. Formeln können mit booleschen Operatoren verknüpft werden. Deskriptoren und Vergleichsoperatoren werden als zweistellige Prädikatenkonstanten unterschiedlicher Sorte eingeführt. Die Argumente stellen die beiden in Beziehung stehenden Objekte dar. Bei den Deskriptoren sind dies z.B. Dokument und Deskriptorwert. Der zweistellige linguistsche Operator "stem" liefert sowohl die Stammformen als auch die Vollformen eines Wortes (z.B.: Stem(x, Bücher) und Stem(Buch, x)).
SimpleStructuredQL ist eine mehrsortige, prädikatenlogische Sprache 2. Stufe. Wenn der λ–Operator für Prädikate (siehe Syntax, Anfragen, Punkt 2) weggelassen würde, wäre SimpleStructuredQL eine mehrsortige, prädikatenloggische Sprache 1. Stufe. SimpleStructuredQL enthält keine Funktionen.

4.1.1 Syntax

Sorten
1. mail, person, host, file, news, menu, unit, string, date, int, regExpr sind Sorten.
2. Das sind alle Sorten.

Alphabet
1. Individuenkonstanten: mail, $mail_1$, ... , person, $person_1$, ... , host, $host_1$, ... , file, $file_1$, ... , news, $news_1$, ... , menu, $menu_1$, ... , unit, $unit_1$, ... , string, $string_1$, ..., date, $date_1$, ..., 1, 2, ...
2. Individuenvariablen verschiedener Sorten: x^{mail}, $x^{mail}{}_1$, ... , x^{person}, $x^{person}{}_1$, ... , x^{host}, $x^{host}{}_1$, ... , x^{file}, $x^{file}{}_1$, ... , x^{news}, $x^{news}{}_1$, ... , x^{menu}, $x^{menu}{}_1$, ... , x^{unit}, $x^{unit}{}_1$, ... , x^{string}, $x^{string}{}_1$, ... , x^{date}, $x^{date}{}_1$, ... , x^{int}, $x^{int}{}_1$, ... , $x^{regExpr}$, $x^{regExpr}{}_1$, ...
3. Prädikatenkonstanten (2-stellig) verschiedener Sorten (wenn nicht explizit angegeben haben die Prädikate die Sorten des jeweils ersten Prädikats der betreffenden Liste, bei a) z.B. die Sorten von Content):
a) Artikel: $Content^{news,string}$, $Date^{news,date}$, Group-id, $Length^{file,int}$, Newsgroup, Subject, Title, User-id, ...
b) Dateien: $Contains^{file,file}$, $Content^{file,string}$, $Date^{file,date}$, $File\text{-}name^{file,string}$, $File\text{-}type^{file,string}$, $Group\text{-}id^{file,string}$, $Length^{file,int}$, $Title^{file,string}$, $User\text{-}id^{file,string}$, ...
c) Dokumenthinweise: $Abstract^{unit,string}$, Creator, Contributors, Coverage, $Date^{unit,date}$, Database, Description, Format, Identifier, Language, Publisher, Relation, Rights, Source, Subject, Type, Title, ...
d) E-Mails: $Action^{mail,string}$, Copy, Content-type, Content-transfer-encoding, Content-id, Content-description, $Date^{mail,date}$, Expires, From, In-reply-to, Message-id, Mime-version, Prece-de(n)ce, Priority, Received, Reply-to, Subject, To, ...
e) Menüs: $Content^{menu,string}$, $Date^{menu,date}$, Group-id, $Length^{file,int}$, Subject, Title, User-id, ...
f) Personen/Gruppen: $Communication\text{-}address\text{-}private^{person,string}$, Communication-address-office, $Date^{person,date}$, Domain-name, Fields-of-business, Information-address, Keywords, Members, Name-private, Name-headquarters, E-Mail-private, E-mail-office, Fax-private, Fax-office, Phone-private, Phone-office, Postal-address-private, Postal-address-office, Personal-data, Public-key, Projects, Title, Type, ...
g) Rechner: $Address\text{-}administrator^{host,string}$, Address-technical-person, $Date\text{-}of\text{-}last\text{-}modification^{host,date}$, E-mail-administrator, E-mail-technical-person, Host-domain-name, Host-type, Information-address, Ip, Name-administrator, Name-technical-person, Nameservers-of-network, Operating-system, Phone-administrator, Phone-technical-person, ...
h) Vergleichsprädikate:

h1) Zahlenvergleiche: $\neq^{date,date}$, $<^{date,date}$, $>^{date,date}$, $\leq^{date,date}$, $\geq^{date,date}$, $\neq^{int,int}$, $<^{int,int}$, $>^{int,int}$, $\leq^{int,int}$, $\geq^{int,int}$

h2) Vergleich von Zeichenfolgen: $Contains^{regExpr,string}$, $Near^{string,int,string}$

h3) Linguistische Operationen: $Upper^{string,string}$, $Lower^{string,string}$, $Stem^{string,string}$, $Fuzzy^{string,string}$, $Soundex^{string,string}$

h4) Thesaurusoperatoren: $Syn^{string,string}$, $NT^{string,string}$, $BT^{string,string}$, $PT^{string,string}$

4. Prädikatenvariablen (2-stellig) verschiedener Sorten: $P^{news,string}{}_1$, $P^{news,string}{}_2$, ... , $P^{file,string}{}_1$, $P^{file,string}{}_2$, ... , $P^{unit,string}{}_1$, $P^{unit,string}{}_2$, ... , $P^{mail,string}{}_1$, $P^{mail,string}{}_2$, ... , $P^{menu,string}{}_1$, $P^{menu,string}{}_2$, ... , $P^{person,string}{}_1$, $P^{person,string}{}_2$, ... , $P^{host,string}{}_1$, $P^{host,string}{}_2$, ..., $P^{string,string}{}_1$, $P^{string,string}{}_2$, ...

5. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation) $\leftrightarrow$ (Äquivalenz)
$\exists$ (es gibt ein), $\forall$ (für alle), λ (alle, die)

6. Technische Symbole: (,)

Formeln

1. Wenn x_1 Individuenkonstante oder -variable der Sorte s_1 ist und x_2 Individuenkonstante oder -variable der Sorte s_2 ist und P eine Prädikatenkonstante oder -variable der Sorten s_1, s_2 ist, dann ist $P^{s1,s2}(x_1, x_2)$ eine Formel.
2. Wenn F, F_1 und F_2 Formeln sind und x^s eine Individuenvariable der Sorte s ist, dann sind auch $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$, $\exists\, x^s\, F$, $\forall x^s\, F$ Formeln.
3. Das sind alle Formeln.

Anfragen

1. Wenn F eine Formel und $x^{s1}{}_1$ ... $x^{sn}{}_n$ Individuenvariablen der Sorten s_1, ..., s_n sind, dann ist $(\lambda\, x^{s1}{}_1 \ldots x^{sn}{}_n)$ F eine Anfrage.
2. Wenn F eine Formel und $P^{s1,\ldots,sn}$ eine Prädikatenvariable der Sorten s_1, ..., s_n ist, dann ist $(\lambda\, P^{s1,\ldots,sn})$ F eine Anfrage.
3. Das sind alle Anfragen.

4.1.2 Semantik

U^{mail} sei eine nichtleere Menge von E-Mails, U^{person} sei eine nichtleere Menge von Personen, U^{host} sei eine nichtleere Menge von Rechnern, U^{file} sei eine nichtleere Menge von Dateien, U^{news} sei eine nichtleere Menge von Artikeln, U^{menu} sei eine nichtleere Menge von Menüs, U^{unit} sei eine nichtleere Menge von Dokumenthinweisen, U^{string} sei eine nichtleere Menge von Zeichenfolgen, U^{date} sei eine nichtleere Menge von Datumseinträgen, U^{int} sei eine nichtleere Menge von positiven, ganzen Zahlen, $U^{regExpr}$ sei eine nichtleere Menge von regulären Ausdrücken.

Interpretation der Individuensymbole

$\Im(news_i) \in U^{news}$, $\Im(file_i) \in U^{file}$, $\Im(unit_i) \in U^{unit}$, $\Im(mail_i) \in U^{mail}$, $\Im(menu_i) \in U^{menu}$, $\Im(person_i) \in U^{person}$, $\Im(host_i) \in U^{host}$, $\Im(string_i) \in U^{string}$, $\Im(date_i) \in U^{date}$, $\Im(regExpr_i) \in U^{regExpr}$, $\Im(1) \in U^{int}$, $\Im(2) \in U^{int}$, ... für i= 1, ..., n (i kann auch weggelassen werden)
$\Im(x^{news}{}_i) \in U^{news}$, $\Im(x^{file}{}_i) \in U^{file}$, $\Im(x^{unit}{}_i) \in U^{unit}$, $\Im(x^{mail}{}_i) \in U^{mail}$, $\Im(x^{menu}{}_i) \in U^{menu}$, $\Im(x^{person}{}_i) \in U^{person}$, $\Im(x^{host}{}_i) \in U^{host}$, $\Im(x^{string}{}_i) \in U^{string}$, $\Im(x^{date}{}_i) \in U^{date}$, $\Im(x^{int}{}_i) \in U^{int}$, $\Im(x^{regExpr}{}_i) \in U^{regExpr}$, ... für i= 1, ..., n (i kann auch weggelassen werden)

Interpretation der Prädikatensymbole

K sei eine Prädikatenkonstante (außer Vergleichsprädikat) und P sei eine Prädikatenvariable.
$\Im(K^{s1,s2}) \subseteq U^{s1} \times U^{s2}$

$\Im(P^{s1,s2}) \subseteq U^{s1} \times U^{s2}$

Interpretation der Formeln

1. x sei Individuenkonstante oder -variable und K sei eine Prädikatenkonstante (außer Vergleichsprädikat)

$\Im$ ist Modell von $K^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $<\Im(x^{s1}_i), \Im(x^{s2}_j)> \in \Im(K^{s1,s2})$

$\Im$ ist Modell von $P^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $<\Im(x^{s1}_i), \Im(x^{s2}_j)> \in \Im(P^{s1,s2})$

2. $\Im$ ist Modell von $\neq^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist ungleich $\Im(x^{s2}_j)$

$\Im$ ist Modell von $<^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist kleiner als $\Im(x^{s2}_j)$

$\Im$ ist Modell von $>^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist größer als $\Im(x^{s2}_j)$

$\Im$ ist Modell von $\leq^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist ist kleiner gleich $\Im(x^{s2}_j)$

$\Im$ ist Modell von $\geq^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist ist größer gleich $\Im(x^{s2}_j)$

$\Im$ ist Modell von $Contains^{regExpr,string}(x^{regExpr}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ enthält den regulären Ausdruck $\Im(x^{regExpr}_1)$

$\Im$ ist Modell von $Near^{string,int,string}(x^{string}_1, x^{int}_2, x^{string}_1)$ gdw
$\Im(x^{string}_3)$ befindet sich in der $\Im(x^{int}_2)$ - Zeichenzahl-Umgebung von $\Im(x^{string}_1)$

$\Im$ ist Modell von $Upper^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist die in Großbuchstaben gewandelte Zeichenfolge von $\Im(x^{string}_1)$

$\Im$ ist Modell von $Lower^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist die in Kleinbuchstaben gewandelte Zeichenfolge von $\Im(x^{string}_1)$

$\Im$ ist Modell von $Stem^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist die Stammform von $\Im(x^{string}_1)$

$\Im$ ist Modell von $Fuzzy^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist rechschreibähnlich zu $\Im(x^{string}_1)$

$\Im$ ist Modell von $Soundex^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist phonetisch ähnlich zu $\Im(x^{string}_1)$

$\Im$ ist Modell von $Syn^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist synonym zu $\Im(x^{string}_1)$

$\Im$ ist Modell von $NT^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist Unterbegriff von $\Im(x^{string}_1)$

$\Im$ ist Modell von $BT^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist Oberbegriff von $\Im(x^{string}_1)$

$\Im$ ist Modell von $PT^{string,string}(x^{string}_1, x^{string}_2)$ gdw
$\Im(x^{string}_2)$ ist Vorzugsbegriff von $\Im(x^{string}_1)$

3. $\Im$ ist Modell von $\neg F$ gdw $\Im$ ist nicht Modell von F.

$\Im$ ist Modell von $F_1 \wedge F_2$ gdw $\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.

$\Im$ ist Modell von $F_1 \vee F_2$ gdw $\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.

$\Im$ ist Modell von $F_1 \rightarrow F_2$ gdw $\Im$ ist Modell von F2 wenn $\Im$ ist Modell von F_1.

$\Im$ ist Modell von $F_1 \leftrightarrow F_2$ gdw $\Im$ ist Modell von F1 gdw $\Im$ ist Modell von F_2.

$\Im$ ist Modell von $\exists\, x^s_i\, F$ gdw $\Im^{Ui}_{xi}$ ist Modell von F für mindestens ein $U_i \in U^s$.

$\Im$ ist Modell von $\forall\, x^s_i\, F$ gdw $\Im^{Ui}_{xi}$ ist Modell von F für alle $U_i \in U^s$.

Interpretation der Anfragen

7. $\Im((\lambda\, x^{s1}_1 \ldots x^{sn}_n)\, F) =$
$\{<U_1, \ldots, U_n>: \Im^{U1\ldots Un}_{x1\ldots xn}$ ist Modell von F für $U_1 \in U^{s1}, \ldots, U_n \in U^{sn}\}$.

ℑ((λ $P^{s1,...,sn}$) F) = {<rel>: $ℑ^{rel}_{P}$ ist Modell von F}.

4.1.3 Beispiele

Individuenbereiche:
U^{news} = {N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8}
U^{unit} = {$UNIT_1$, $UNIT_2$}
U^{file} = {F_1, F_2, F_3, F_4, F_5, F_6}
U^{mail} = {M_1, M_2, M_3, M_4, M_5}
U^{string} = {"comp.lang.java.programmer", "miller@cs.mit.edu", ...}
$U^{regExpr}$ = {"*ormation retrieval", "c?t", ...}
Bemerkung: Es werden in U^{string} und $U^{regExpr}$ nur einige Zeichenfolgen beispielhaft aufgeführt.

Relationen:
NEWSGROUP = ℑ (Newsgroup^{news,string}) =
{<N_1, "comp.lang.java.programmer">, <N_2, "comp.lang.java.programmer">,
<N_3, "comp.lang.java.programmer">, <N_4, "comp.lang.java.programmer">,
<N_5, "comp.lang.java.programmer">, <N_6, "comp.lang.java.programmer">}
USER-ID$^{news \times string}$ = ℑ (User-id^{news,string}) =
{<N_1, "meier@cs.mit.edu">, <N_2, "miller@cs.mit.edu">, <N_3, "lewis@cs.mit.edu">,
<N_4, "smith@cs.mit.edu">, <N_5, "dalitz@zib.de">, <N_6, "bene@cs.tu-berlin.de">}
DATE$^{news \times date}$ = ℑ (Date^{news,date}) =
{<N_1, 1.1.1998>, <N_2, 1.1.1998>, <N_3, 1.7.1997>, <N_4, 1.1.1997>,
<N_5, 1.1.1997>, <N_6, 1.1.1997>, <N_7, 1.1.1998>, <N_8, 1.1.1998>}
CONTENT$^{news \times string}$ = ℑ (Content^{news,string}) =
{<N_1, "Information Retrieval has to move ...">, <N_2, "No, Retrieval has to ...">,
<N_3, "/* Dies ist eine Methode zur Stringverarbeitung ...">,
<N_4, "/* Zeichenfolgensuche auf Basis des Match-Algorithmus ...">}
SUBJECT = ℑ (Subject^{unit,string}) =
{<$UNIT_1$, "information">, <$UNIT_1$, "retrieval">, <$UNIT_2$, "internet">}
DATE$^{unit \times date}$ = ℑ (Date^{unit,date}) = {<$UNIT_1$, 12.12.2000>}
LANGUAGE = ℑ (Language^{unit,string}) = {<$UNIT_1$, "German">, <$UNIT_2$, "German">}
DATABASE = ℑ (Database^{unit,string}) =
{<$UNIT_1$, "Loc-Books">, <$UNIT_2$, " Loc-Books ">}
FILENAME = ℑ (File-name^{file,string}) =
{<F_1, "ftp.cs.tu-berlin.de/pub">, <F_2, "INDEX">, <F_3, "audio">, <F_4, "gnu">, <F_5, "net">,
<F_6, "test.txt">, <F_7, "test.txt">}
USERID$^{file \times string}$ = ℑ (User-id^{file,string}) =
{<F_1, "erhard@cs.tu-berlin.de">, <F_2, "erhard@cs.tu-berlin.de">}
CONTAINS = ℑ (Contains^{file,file}) =
{<F_1, F_2>, <F_1, F_3>, <F_1, F_4> ,<F_1, F_5>, <F_4, F_6>, <F_5, F_7>}
COPY = ℑ (Copy^{mail,string}) =
{<M_1, "bene@cs.tu-berlin.de">, <M_3, " miller@cs.mit.edu ">}
DATE$^{mail \times date}$ = ℑ (Date^{mail,date}) =
{<M_1, 1.1.1998>, <M_2, 1.1.1998>, <M_3, 1.7.1997>, <M_4, 1.1.1997>, <M_5, 1.1.1997>}

TO = ℑ (To^{mail,string}) =
{<M_1, "josefw@cs.tu-berlin.de">, <M_2, "lewis@cs.mit.edu">, <M_3, "lewis@cs.mit.edu">,
<M_4, "dalitz@zib.de">, <M_5, "smith@cs.mit.edu">}

$\mathrm{FROM} = \mathfrak{I}\,(\mathrm{From}^{mail,string}) =$

$\{\langle M_1, \text{"erhard@cs.tu-berlin.de"}\rangle, \langle M_2, \text{"smith@cs.mit.edu"}\rangle, \langle M_3, \text{"smith@cs.mit.edu"}\rangle,$
$\langle M_4, \text{"lügger@zib.de"}\rangle, \langle M_5, \text{"lewis@cs.mit.edu"}\rangle\}$

Bemerkung: Es werden nur bei den doppeldeutigen Relationen die Sorten explizit hochgestellt.

Frage 1: Welche Artikel liegen in der Newsgroup "comp.lang.java.programmer" ?

$\mathfrak{I}((\lambda\, x^{news})\ \mathrm{Newsgroup}^{news,string}(x^{news}, \text{"comp.lang.java.programmer"})) =$
$\{\langle U\rangle : \mathfrak{I}^U{}_x \text{ ist Modell von } \mathrm{Newsgroup}^{news,string}(x^{news}, \text{"comp.lang.java.programmer"})$
$\text{für } U \in \mathcal{U}^{string}\} =$
$\{\langle U\rangle : \langle\mathfrak{I}^U{}_x(x^{news}), \mathfrak{I}^U{}_x(\text{"comp.lang.java.programmer"})\rangle \in \mathfrak{I}^U{}_x\,(\mathrm{Newsgroup}^{news,string})$
$\text{für } U \in \mathcal{U}^{string}\} =$
$\{\langle U\rangle : \langle\mathfrak{I}^U{}_x(U), \text{"comp.lang.java.programmer"}\rangle \in \mathrm{NEWSGROUP} \text{ für } U \in \mathcal{U}^{string}\} =$
$\{\langle N_1\rangle, \langle N_2\rangle, \langle N_3\rangle, \langle N_4\rangle, \langle N_5\rangle, \langle N_6\rangle\}$

Frage 2: Von was handelt das Dokument unit_1 ?

$\mathfrak{I}((\lambda\, x^{string})\ \mathrm{Subject}^{unit,string}\,(\mathrm{unit}_1, x^{string})) =$
$\{\langle U\rangle : \mathfrak{I}^U{}_x \text{ ist Modell von } \mathrm{Subject}^{unit,string}\,(\mathrm{unit}_1, x^{string}) \text{ für } U \in \mathcal{U}^{string}\} =$
$\{\langle U\rangle : \langle\mathfrak{I}^U{}_x\,(\mathrm{unit}_1), \mathfrak{I}^U{}_x(x^{string})\rangle \in \mathfrak{I}^U{}_x\,(\mathrm{Subject}^{unit,string}) \text{ für } U \in \mathcal{U}^{string}\} =$
$\{\langle U\rangle : \langle \mathrm{UNIT}_1, U\rangle \in \mathrm{SUBJECT} \text{ für } U \in \mathcal{U}^{string}\} =$
$\{\langle\text{"information"}\rangle, \langle\text{"retrieval"}\rangle\}$

Frage 3: Welche Dokumente liegen in der Datenbank "LoC-Books" ?

$\mathfrak{I}((\lambda\, x^{unit})\ \mathrm{Database}^{unit,string}(x^{unit}, \text{"LoC-Books"})) =$
$\{\langle U\rangle : \mathfrak{I}^U{}_x \text{ ist Modell von } \mathrm{Database}^{unit,string}(x^{unit}, \text{"LoC-Books"}) \text{ für } U \in \mathcal{U}^{string}\} =$
$\{\langle U\rangle : \langle\mathfrak{I}^U{}_x\,(x^{unit}), \mathfrak{I}^U{}_x(\text{"LoC-Books"})\rangle \in \mathfrak{I}^U{}_x\,(\mathrm{Database}^{unit,string}) \text{ für } U \in \mathcal{U}^{string}\} =$
$\{\langle U\rangle : \langle U, \text{"LoC-Books"}\rangle \in \mathrm{DATABASE} \text{ für } U \in \mathcal{U}^{string}\} =$
$\{\langle \mathrm{UNIT}_1\rangle, \langle \mathrm{UNIT}_2\rangle\}$

Frage 4: Welche Artikel aus der Newsgroup "comp.lang.java.programmer" stammen von "meier@cs.mit.edu" oder "miller@cs.mit.edu" und sind am 1.1.1998 veröffentlicht worden ?

$\mathfrak{I}((\lambda x^{news})\ \ \mathrm{Newsgroup}^{news,string}(x^{news}, \text{"comp.lang.java.programmer"}) \wedge$
$(\mathrm{User\text{-}id}^{news,string}(x^{news}, \text{"meier@cs.mit.edu"}) \vee$
$\mathrm{User\text{-}id}^{news,string}(x^{news}, \text{"miller@cs.mit.edu"})) \wedge$
$\mathrm{Date}^{news,date}(x^{news}, 1.1.1998)$
$\{\langle U\rangle : \mathfrak{I}^U{}_x \text{ ist Modell von}$
$\mathrm{Newsgroup}^{news,string}(x^{news}, \text{"comp.lang.java.programmer"}) \wedge$
$(\mathrm{User\text{-}id}^{news,string}(x^{news}, \text{"meier@cs.mit.edu"}) \vee$
$\mathrm{User\text{-}id}^{news,string}(x^{news}, \text{"miller@cs.mit.edu"})) \wedge$
$\mathrm{Date}^{news,date}(x^{news}, 1.1.1998) \text{ für } U \in \mathcal{U}^{news}\} =$
$\{\langle U\rangle : \mathfrak{I}^U{}_x \text{ ist Modell von}$
$(\mathrm{Newsgroup}^{news,string}(x^{news}, \text{"comp.lang.java.programmer"}) \wedge$
$(\mathrm{User\text{-}id}^{news,string}(x^{news}, \text{"meier@cs.mit.edu"}) \vee$
$\mathrm{User\text{-}id}^{news,string}(x^{news}, \text{"miller@cs.mit.edu"})) \wedge$
$\mathrm{Date}^{news,string}(x^{news}, 1.1.1998)) \text{ für } U \in \mathcal{U}^{news}\} =$
$\{\langle U\rangle : \langle\mathfrak{I}^U{}_x(x^{news})\ ,\mathfrak{I}^U{}_x(\text{"comp.lang.java.programmer"})\rangle \in$
$\mathfrak{I}^U{}_x(\mathrm{Newsgroup}^{news,string}) \text{ und}$

$(<\mathfrak{I}^{U}_{x}(x^{news}), \mathfrak{I}^{U}_{x}(\text{"meier@cs.mit.edu"})> \in \mathfrak{I}^{U}_{x}(\text{User-id}^{news,string})$ oder
$(<\mathfrak{I}^{U}_{x}(x^{news}), \mathfrak{I}^{U}_{x}(\text{"miller@cs.mit.edu"})> \in \mathfrak{I}^{U}_{x}(\text{User-id}^{news,string}))$ und
$<\mathfrak{I}^{U}_{x}(x^{news}), \mathfrak{I}^{U}_{x}(1.1.1998)> \in \mathfrak{I}^{U}_{x}(\text{Date}^{news,string})$ für $U \in U^{news}\}=$
$\{<U> : <U, \text{"comp.lang.java.programmer"}> \in$ NEWSGROUP und
$(<U, \text{"meier@cs.mit.edu"}> \in \text{USERID}^{news\times string}$ oder
$(<U, \text{"miller@cs.mit.edu"}> \in \text{USERID}^{news\times string})$ und
$<U, 1.1.1998> \in \text{DATE}^{news\times date}$ für $U \in U^{news}\}=$
$\{<N_1>, <N_2>\}$

Frage 5: Welche Dateien liegen im Verzeichnis "ftp.cs.tu-berlin.de/pub" ?
$\mathfrak{I}((\lambda x^{file}_{1})\ \exists x^{file}_{2}\ \text{File-name}^{file,string}(x^{file}_{2}, \text{"ftp.cs.tu-berlin.de/pub"}) \wedge$
$\text{Contains}^{file,file}(x^{file}_{2}, x^{file}_{1}))$
$\{<U_1> : \mathfrak{I}^{U1}_{x1}$ ist Modell von
$\exists x^{file}_{2}\ \text{File-name}^{file,string}(x^{file}_{2}, \text{"ftp.cs.tu-berlin.de/pub"}) \wedge$
$\text{Contains}^{file,file}(x^{file}_{2}, x^{file}_{1})$ für $U_1 \in U^{file}\}=$
$\{<U_1> : \mathfrak{I}^{U1\ U2}_{x1\ x2}$ ist Modell von
$(\text{File-name}^{file,string}(x^{file}_{2}, \text{"ftp.cs.tu-berlin.de/pub"}) \wedge$
$\text{Contains}^{file,file}(x^{file}_{2}, x^{file}_{1}))$ für $U_1 \in U^{news}$ und für mindestens ein $U_2 \in U^{file}\} =$
$\{<U_1> : <\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{file}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}(\text{"ftp.cs.tu-berlin.de/pub"})> \in$
$\mathfrak{I}^{U1\ U2}_{x1\ x2}(\text{File-name}^{file,string})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{file}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{file}_{1})> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(\text{Contains}^{file,file})$
für $U_1 \in U^{news}$ und für mindestens ein $U_2 \in U^{file}\}=$
$\{<U_1> : <U_2, \text{"ftp.cs.tu-berlin.de/pub"}> \in$ FILENAME und
$<U_2, U_1> \in$ CONTAINS
für $U_1 \in U^{news}$ und für mindestens ein $U_2 \in U^{file}\}=$
$\{<F_2>, <F_3>, <F_4>, <F_5>\}$

Frage 6: Welche Artikel aus der Newsgroup "comp.lang.java.programmer" enthalten im "Content" den regulären Ausdruck "*formation retrieval" ?
$\mathfrak{I}((\lambda x^{news}_{1})\ \exists x^{string}_{2}\ \text{Newsgroup}^{news,string}(x^{news}_{1}, \text{"comp.lang.java.programmer"}) \wedge$
$\text{Content}^{news,string}(x^{news}_{1}, x^{string}_{2}) \wedge \text{Contains}^{regExpr,string}(\text{"*formation retrieval"}, x^{string}_{2}) =$
$\{<U_1> : \mathfrak{I}^{U1}_{x1}$ ist Modell von
$\exists x^{string}_{2}\ \text{Newsgroup}^{news,string}(x^{news}_{1}, \text{"comp.lang.java.programmer"}) \wedge$
$\text{Content}^{news,string}(x^{news}_{1}, x^{string}_{2}) \wedge$
$\text{Contains}^{regExpr,string}(\text{"*formation retrieval"}, x^{string}_{2})$ für $U_1 \in U^{news}\} =$
$\{<U_1> : \mathfrak{I}^{U1\ U2}_{x1\ x2}$ ist Modell von
$\text{Newsgroup}^{news,string}(x^{news}_{1}, \text{"comp.lang.java.programmer"}) \wedge$
$\text{Content}^{news,string}(x^{news}_{1}, x^{string}_{2}) \wedge \text{Contains}^{regExpr,string}(\text{"*formation retrieval"}, x^{string}_{2})$
für $U_1 \in U^{news}$ und für mindestens ein $U_2 \in U^{string}\} =$
$\{<U_1>:<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{news}_{1}), \mathfrak{I}^{U1\ U2}_{x1\ x2}(\text{"comp.lang.java.programmer"})> \in$
$\mathfrak{I}^{U1\ U2}_{x1\ x2}(\text{Newsgroup}^{news,string})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{news}_{1}), \mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{string}_{2})> \in$
$\mathfrak{I}^{U1\ U2}_{x1\ x2}(\text{Content}^{news,string})$ und
$\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{string}_{2})$ enthält den regulären Ausdruck $\mathfrak{I}^{U1\ U2}_{x1\ x2}(\text{"*formation retrieval"})$
für $U_1 \in U^{news}$ und für mindestens ein $U_2 \in U^{string}\}=$
$\{<U_1>:<U_1, \text{"comp.lang.java.programmer"}> \in$ NEWSGROUP und
$<U_1, U_2> \in \text{CONTENT}^{news\times string}$ und

U_2 enthält den regulären Ausdruck "*formation retrieval"
für $U_1 \in U^{news}$ und für mindestens ein $U_2 \in U^{string}\}=$

$\{<N_1>\}$

Frage 7: Welche Deskriptoren werden für E-Mails benutzt?

$\mathfrak{I}((\lambda\ P^{mail,string}{}_1)\ \exists\ x^{mail}{}_1\ \exists\ x^{string}{}_2\ (P^{mail,string}{}_1(x^{mail}{}_1,\ x^{string}{}_2))) =$

$\{<rel> : \mathfrak{I}^{rel}{}_P$ ist Modell von $\exists\ x^{mail}{}_1\ \exists\ x^{string}{}_2\ (P^{mail,string}{}_1(x^{mail}{}_1,\ x^{string}{}_2))\}=$

$\{<rel> : \mathfrak{I}^{rel\ U1\ U2}{}_{P\ x1\ x2}$ ist Modell von $P^{mail,string}{}_1(x^{mail}{}_1,\ x^{string}{}_2)$ für mindestens ein $U_1 \in U^{mail}$ und ein $U_2 \in U^{string}\}$

$\{<rel> : <\mathfrak{I}^{rel\ U1\ U2}{}_{P\ x1\ x2}\ (x^{mail}{}_1),\ \mathfrak{I}^{rel\ U1\ U2}{}_{P\ x1\ x2}(x^{string}{}_2)> \in \mathfrak{I}^{rel\ U1\ U2}{}_{P\ x1\ x2}(P^{mail,string}{}_1)$ für mindestens ein $U_1 \in U^{mail}$ und ein $U_2 \in U^{string}\}$

$\{<rel> : <U_1,\ U_2> \in \mathfrak{I}^{rel}{}_P(P^{mail,string}{}_1)$ für mindestens ein $U_1 \in U^{mail}$ und ein $U_2 \in U^{string}\}$

$\{<COPY>,\ <FROM>,\ <TO>,\ <DATE^{mail \times date}>\}$

Die folgenden Anfragen werden aus Reiner (1991) S. 87ff übernommen. Für deren Interpretation gelten dieselben Individuenbereiche wie bei Reiner. Statt der bei Reiner verwendeten Prädikatenkonstanten AU, PY, DT, DE werden die Prädikatenkonstanten Author, Date, Type, Subject benutzt.

Frage 8: Welche Dokumente handeln von Schlange?

$\mathfrak{I}((\lambda\ x^{unit})\ Subject^{unit,string}(x^{unit},\ \text{"Schlange"})) =$

$\{<U> : \mathfrak{I}^{U}{}_x$ ist Modell von $Subject^{unit,string}\ (x^{unit},\ \text{"Schlange"})$ für $U \in U^{unit}\} =$

$\{<U> : <\mathfrak{I}^{U}{}_x\ (x^{unit}),\ \mathfrak{I}^{U}{}_x(\text{"Schlange"}) > \in \mathfrak{I}^{U}{}_x\ (Subject^{unit,string})$ für $U \in U^{unit}\} =$

$\{<U> : <U,\ \text{"Schlange"}> \in$ SUBJECT für $U \in U^{unit}\} =$

$\{<D_1>,\ <D_2>,\ <D_3>,\ <D_4>,\ <D_9>,\ <D_{10}>,\ <D_{18}>\}$

Frage 9: Welche Dokumente handeln nur von Schlange?

$\mathfrak{I}((\lambda\ x^{unit})\ \ Subject^{unit,string}(x^{unit},\ \text{"Schlange"}) \wedge$
$\neg\ Subject^{unit,string}(x^{unit},\ \text{"Aberglaube"}) \wedge$
$\neg\ Subject^{unit,string}(x^{unit},\ \text{"Affe"}) \wedge ... \wedge$
$\neg\ Subject^{unit,string}(x^{unit},\ \text{"Zucht"})\ \ ) =$

$\{<U> : \mathfrak{I}^{U}{}_{x1}$ ist Modell von ($Subject^{unit,string}(x^{unit},\ \text{"Schlange"}) \wedge$
$\neg\ Subject^{unit,string}(x^{unit},\ \text{"Aberglaube"}) \wedge$
$\neg\ Subject^{unit,string}(x^{unit},\ \text{"Affe"}) \wedge ... \wedge$
$\neg\ Subject^{unit,string}(x^{unit},\ \text{"Zucht"})$) für $U \in U^{unit}\} =$

$\{<U> : <\mathfrak{I}^{U}{}_x(x^{unit}),\ \mathfrak{I}^{U}{}_x(\text{"Schlange"})> \in \mathfrak{I}^{U}{}_x(Subject^{unit,string})$ und
$<\mathfrak{I}^{U}{}_x(x^{unit}),\ \mathfrak{I}^{U}{}_x(\text{"Aberglaube"})> \notin \mathfrak{I}^{U}{}_x(Subject^{unit,string})$ und
$<\mathfrak{I}^{U}{}_x(x^{unit}),\ \mathfrak{I}^{U}{}_x(\text{"Affe"})> \notin \mathfrak{I}^{U}{}_x(Subject^{unit,string})$ und ... und
$<\mathfrak{I}^{U}{}_x(x^{unit}),\ \mathfrak{I}^{U}{}_x(\text{"Zucht"})> \notin \mathfrak{I}^{U}{}_{x1}\ (Subject^{unit,string})$ für $U \in U^{unit}\}=$

$\{<U_1> : <U,\ \text{"Schlange"}> \in$ SUBJECT und
$<U,\ \text{"Aberglaube"}> \notin$ SUBJECT und
$<U,\ \text{"Affe"}> \notin$ SUBJECT und ... und
$<U,\ \text{"Zucht"}> \notin$ SUBJECT für $U \in U^{unit}\}=$

$\{<D_{18}>\}$

Frage 10: Welche Dokumente sind außerhalb der Jahre 1960-1980 veröffentlicht worden?

$\mathfrak{I}((\lambda\ x^{unit}{}_1)\ \exists\ x^{date}{}_2\ \ \ \ Date^{unit,date}\ (x^{unit}{}_1,\ x^{date}{}_2) \wedge \leq^{date,date}(\ x^{date}{}_2,\ \text{"1960"}) \wedge$

$\geq^{date,date}(x^{date}_{2}, "1980"))$

$\{< U_1> : \mathfrak{I}^{U1}_{x1}$ ist Modell von
$(\exists\, x^{date}_{2}\ Date^{unit,date}(x^{unit}_{1}, x^{date}_{2}) \wedge \leq^{date,date}(x^{date}_{2}, "1960") \wedge \geq^{date,date}(x^{date}_{2}, "1980"))$
für $U_1 \in \mathcal{U}^{unit}\}$

$\{< U_1> : \mathfrak{I}^{U1\ U2}_{x1\ x2}$ ist Modell von
$(Date^{unit,date}(x^{unit}_{1}, x^{date}_{2}) \wedge \leq^{date,date} (x^{date}_{2}, "1960") \wedge \geq^{date,date} (x^{date}_{2}, "1980"))$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{date}\}$

$\{< U_1> :\quad <\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{unit}_{1}), \mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{date}_{2})> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(Date^{unit,date})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{date}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}("1960")> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(\leq^{date,date})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{date}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}("1980")> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(\geq^{date,date})$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{date}\}$

$\{< U_1> :\quad <U_1, U_2> \in DATE^{unit\times date}$ und $<U_2, "1960"> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(\leq^{date,date})$ und
$< U_2, "1980"> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(\geq^{date,date})$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{date}\}$

$\{<D_1>, <D_4>, <D_9>, <D_{12}>, <D_{13}>, <D_{14}>, <D_{15}>, <D_{16}>\}$

Frage 11: In welchem Jahr hat Schulte, R. ein Buch geschrieben?

$\mathfrak{I}((\lambda\, x^{date}_{1})\ \exists x^{unit}_{2}\quad Author^{unit,string}(x^{unit}_{2}, "Schulte, R.") \wedge Date^{unit,date}(x^{unit}_{2}, x^{date}_{1}) \wedge$
$Type^{unit,string}(x^{unit}_{2}, "Buch")) =$

$\{< U_1> : \mathfrak{I}^{U1}_{x1}$ ist Modell von
$(\exists x^{unit}_{2}\ Author^{unit,string}(x^{unit}_{2}, "Schulte, R.") \wedge Date^{unit,date}(x^{unit}_{2}, x^{date}_{1}) \wedge$
$Type^{unit,string}(x^{unit}_{2}, "Buch"))$ für $U_1 \in \mathcal{U}^{unit}\}$

$\{< U_1> : \mathfrak{I}^{U1\ U2}_{x1\ x2}$ ist Modell von
$(Author^{unit,string}(x^{unit}_{2}, "Schulte, R.") \wedge Date^{unit,date}(x^{unit}_{2}, x^{date}_{1}) \wedge Type^{unit,string}(x^{unit}_{2}, "Buch"))$

für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\}$

$\{< U_1> : <\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{unit}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}("Schulte, R.")> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(Author^{unit,string})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{unit}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{date}_{1})> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(Date^{unit,date})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{unit}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}("Buch")> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(Type^{unit,string})$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\}$

$\{< U_1> : <U_2, "Schulte, R."> \in AUTHOR$ und $<U_1, U_2> \in DATE^{unit\times date}$ und
$< U_1, "Buch"> \in TYPE^{unit\times string}$
für $U_1 \in \mathcal{U}^{unit}$ für mindestens ein $U_2 \in \mathcal{U}^{unit}\}$

$\{<"1980">\}$

Frage 12: Welches sind die Co-Autoren von Freitag, G.E.?

$\mathfrak{I}((\lambda\, x^{string}_{1})\ \exists\, x^{unit}_{2}\ Author^{unit,string}(x^{unit}_{2}, "Freitag, G.E.") \wedge Author^{unit,string}(x^{unit}_{2}, x^{string}_{1}) \wedge$
$\neq^{string,string}(x^{string}_{1}, "Freitag, G.E.")) =$

$\{<U_1> : \mathfrak{I}^{U1}_{x1}$ ist Modell von $\exists\, x^{unit}_{2}\ Author^{unit,string}(x^{unit}_{2}, "Freitag, G.E.") \wedge$
$Author^{unit,string}(x^{unit}_{2}, x^{string}_{1}) \wedge \neq^{string,string}(x^{string}_{1}, "Freitag, G.E."))$ für $U_1 \in \mathcal{U}^{unit}\} =$

$\{<U_1> : \mathfrak{I}^{U1\ U2}_{x1\ x2}$ ist Modell von $(Author^{unit,string}(x^{unit}_{2}, "Freitag, G.E.") \wedge$
$Author^{unit,string}(x^{unit}_{2}, x^{string}_{1}) \wedge \neq^{string,string}(x^{string}_{1}, "Freitag, G.E."))$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\}=$

$\{<U_1> : <\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{unit}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}("Freitag, G.E.")> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(Author^{unit,string})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{unit}_{2}), \mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{string}_{1})> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(Author^{string,string})$ und
$<\mathfrak{I}^{U1\ U2}_{x1\ x2}(x^{string}_{1}), \mathfrak{I}^{U1\ U2}_{x1\ x2}("Freitag, G.E.")> \in \mathfrak{I}^{U1\ U2}_{x1\ x2}(\neq^{string,string})$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\}$

$\{<U_1> : < U_2$, "Freitag, G.E."$> \in$ AUTHOR und
$< U_2, U_1> \in$ AUTHOR und $< U_1$, "Freitag, G.E."$> \in \Im^{U1\ U2}{}_{x1\ x2}(\neq^{string,string})$
für $U_1 \in U^{unit}$ und für mindestens ein $U_2 \in U^{unit}\}$

{<"Deckert, K.">, <"Grzimek, B.">, <"Günther, K.">, <"Heusser, H.R.">, <"Kuhn, O.">, <"Peters, G.">, <"Sterba, G.">, <"Thenius, E.">}

4.2 SimpleLinkQL: Eine Anfragesprache für einfach link-strukturierte Dokumente

Anwender von SimpleLinkQL sind Benutzer die einen Zugriff auf einfache link-strukturierte Dokumente wünschen. SimpleLinkQL enthält Sprachelemente für Dokumente (Individuenkonstanten und variablen) und für zweistellige Beziehungen zwischen diesen (die Prädikatenkonstante Conn). Formeln können mit booleschen Operatoren verknüpft werden. Dokumente werden als einfach (unstrukturiert) vorausgesetzt. Beziehungen zwischen Dokumenten werden nicht weiter klassifiziert (z.B. mit Namen von Beziehungen oder mit Eigenschaften von Beziehungen). Beziehungen sind gerichtet. Die Bestände können Zyklen enthalten.
SimpleLinkQL ist eine prädikatenlogische Sprache 1. Stufe. Sie enthält keine Funktionen.

4.2.1 Syntax

Alphabet

1. Individuenkonstanten für Dokumente: unit, $unit_1$, $unit_2$, ...
2. Individuenvariablen für Dokumente: x, x_1, x_2, ...
3. Prädikatenkonstante (2-stellig): Conn
4. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation) $\leftrightarrow$ (Äquivalenz)
$\exists$ (es gibt ein), $\forall$ (für alle), λ (alle, die)
5. Technische Symbole: (,)

Formeln

1. $Conn(x_i, x_j)$ sind Formeln.
2. Wenn F, F_1 und F_2 Formeln sind, dann auch $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$
$\exists x_i F$, $\forall x_i F$.
3. Das sind alle Formeln.

Anfragen

1. Wenn F eine Formel ist, dann ist $(\lambda\ x_1 \ldots x_n)$ F eine Anfrage.
2. Das sind alle Anfragen.

4.2.2 Semantik

U^{unit} ist eine nichtleere Menge von elementaren Dokumenten.

Interpretation der nichtlogischen Symbole

$\Im(unit) \in U^{unit}$, $\Im(unit_1) \in U^{unit}$, ... , $\Im(x) \in U^{unit}$, $\Im(x_1) \in U^{unit}$, ...
$\Im(Conn) \subseteq U^{unit} \times U^{unit}$

Formeln

1. $\Im$ ist Modell von $Conn(x_i, x_j)$ gdw $<\Im(x_i), \Im(x_j)> \in \Im(Conn)$.
2. $\Im$ ist Modell von $\neg F$ gdw $\Im$ ist nicht Modell von F.
$\Im$ ist Modell von $F_1 \wedge F_2$ gdw $\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.

$\Im$ ist Modell von $F_1 \vee F_2$ gdw $\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $F_1 \rightarrow F_2$ gdw $\Im$ ist Modell von F2 wenn $\Im$ ist Modell von F_1.
$\Im$ ist Modell von $F_1 \leftrightarrow F_2$ gdw $\Im$ ist Modell von F1 gdw $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $\exists\, x_i$ F gdw $\Im^{Ui}{}_{xi}$ ist Modell von F für mindestens ein $U_i \in U^{unit}$.
$\Im$ ist Modell von $\forall\, x_i$ F gdw $\Im^{Ui}{}_{xi}$ ist Modell von F für alle $U_i \in U^{unit}$.

Anfragen
$\Im(\lambda\, x_1 \ldots x_n)\, F) = \{<U_1, \ldots, U_n>: \Im^{U1 \ldots Un}{}_{x1 \ldots xn}$ ist Modell von F für $U_i \in U^{unit}\}$

4.2.3 Beispiele

Individuenbereich:
$UNIT_1$- $UNIT_{11}$ werden durch ihre URL definiert:
$UNIT_1$= "http://www.cs.tu-berlin.de/~josefw/phd/index.html"
$UNIT_2$ = "http://www.cs.tu-berlin.de/~josefw/phd/introduction.html"
$UNIT_3$ = "http://www.cs.tu-berlin.de/~josefw/phd/state.html"
$UNIT_4$ = "http://www.cs.tu-berlin.de/~josefw/phd/query.html"
$UNIT_5$ = "http://www.cs.tu-berlin.de/~josefw/phd/hynternetQL.html"
$UNIT_6$ = "http://www.cs.tu-berlin.de/~josefw/phd/literature.html"
$UNIT_7$ = "http://www.cs.tu-berlin.de/~josefw/phd/literature/Abiteboul,Beeri1995.html"
$UNIT_8$ = "http://www.cs.tu-berlin.de/~josefw/phd/literature/Ackermann,Hilbert1972.html"
$UNIT_9$ = "http://www.cs.tu-berlin.de/~josefw/phd/literature/Afrati,Koutras1990.html"
$UNIT_{10}$ = "http://www.cs.huji.ac.il/~beeri"
$UNIT_{11}$ = "http://www-rocq.inria.fr/~abitebou/pub/icdt97.semistructured.talk.ps"

U^{unit} = $\{UNIT_1, UNIT_2, UNIT_3, UNIT_4, UNIT_5, UNIT_6, UNIT_7, UNIT_8, UNIT_9, UNIT_{10}, UNIT_{11}\}$

Relation:
CONN = $\Im$ (Conn) = $\{<UNIT_1, UNIT_2>, <UNIT_1, UNIT_3>, <UNIT_1, UNIT_4>, <UNIT_1, UNIT_5>, <UNIT_1, UNIT_6>, <UNIT_6, UNIT_7>, <UNIT_6, UNIT_8>, <UNIT_6, UNIT_9>, <UNIT_1, UNIT_7>, <UNIT_9, UNIT_9>, <UNIT_7, UNIT_{10}>, <UNIT_7, UNIT_{11}>\}$

Die Beziehungen stellen sich als Graph folgendermaßen dar:

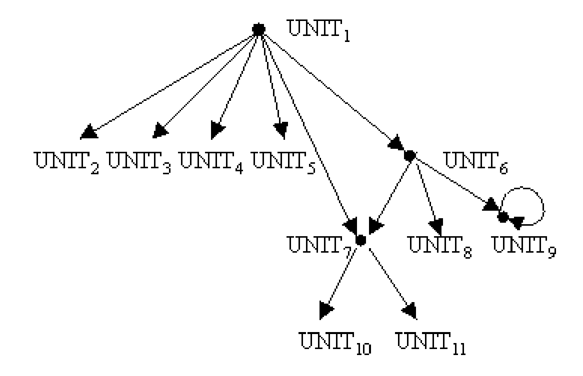


Abbildung 36: Graphische Darstellung der Links

Bemerkungen: $UNIT_6$ und $UNIT_7$ haben denselben Vaterknoten. Es existiert eine zyklische Beziehung bei $UNIT_9$. $UNIT_7$ hat zwei Vorgänger: $UNIT_1$ und $UNIT_6$.

Frage 1: Welche Dokumente stehen zum Dokument $unit_1$ in einer Beziehung?

$\mathfrak{I}((\lambda\ x)\ Conn(unit_1, x)) =$
$\{\langle U\rangle: \mathfrak{I}^U{}_x$ ist Modell von $Conn(unit_1, x)\} =$
$\{\langle U\rangle: \langle\mathfrak{I}^U{}_x(unit_1), \mathfrak{I}^U{}_x(x)\rangle \in \mathfrak{I}^U{}_x(Conn)$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle U\rangle: \langle UNIT_1, U\rangle \in CONN$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle UNIT_2\rangle, \langle UNIT_3\rangle, \langle UNIT_4\rangle, \langle UNIT_5\rangle, \langle UNIT_6\rangle, \langle UNIT_7\rangle\}$

Frage 2: Welche Dokumente stehen zu Dokumenten in Beziehung, die zum Dokument $unit_1$ eine Beziehung haben ?

$\mathfrak{I}((\lambda\ x_1)\ (\exists\ x_2\ Conn(x_2, x_1) \wedge Conn(unit_1, x_2)) =$
$\{\langle U_1\rangle : \mathfrak{I}^{U1}{}_{x1}$ ist Modell von $\exists\ x_2\ Conn(x_2, x_1) \wedge\ Conn(unit_1, x_2)\} =$
$\{\langle U_1\rangle : \mathfrak{I}^{U1\ U2}{}_{x1\ x2}$ ist Modell von $Conn(x_2, x_1) \wedge\ Conn(unit_1, x_2)$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\} =$
$\{\langle U_1\rangle : \langle\mathfrak{I}^{U1\ U2}{}_{x1\ x2}(x_2), \mathfrak{I}^{U1\ U2}{}_{x1\ x2}\ (x_1)\rangle \in \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(Conn)$ und
$\langle\mathfrak{I}^{U1\ U2}{}_{x1\ x2}(unit_1), \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(x_2)\rangle \in \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(Conn)$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\} =$
$\{\langle U_1\rangle : \langle U_2, U_1\rangle \in CONN$ und $\langle UNIT_1, U_2\rangle \in CONN$
für $U_1 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\} =$
$\{\langle UNIT_7\rangle, \langle UNIT_8\rangle, \langle UNIT_9\rangle, \langle UNIT_{10}\rangle, \langle UNIT_{11}\rangle\}$

Frage 3: Welche Dokumente stehen nicht zum Dokument $unit_1$ in einer Beziehung ?

$\mathfrak{I}((\lambda\ x) \neg\ Conn(unit_1, x)) =$
$\{\langle U\rangle : \mathfrak{I}^U{}_x$ ist nicht Modell von $Conn(unit_1, x)$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle U\rangle : \langle\mathfrak{I}^U{}_x(unit_1), \mathfrak{I}^U{}_x(x)\rangle \notin \mathfrak{I}^U{}_x(Conn)$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle U\rangle : \langle UNIT_1, U\rangle \notin CONN$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle UNIT_8\rangle, \langle UNIT_9\rangle, \langle UNIT_{10}\rangle, \langle UNIT_{11}\rangle\}$

Frage 4: Welche Dokumente stehen zum Dokument $unit_1$ oder $unit_6$ in Beziehung?

$\mathfrak{I}((\lambda\ x)\ Conn(unit_1, x) \vee Conn(unit_6, x)) =$
$\{\langle U\rangle : \mathfrak{I}^U{}_x$ ist Modell von $Conn(unit_1, x) \vee\ Conn(unit_6, x)$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle U\rangle : \langle\mathfrak{I}^U{}_x(unit_1), \mathfrak{I}^U{}_x(x)\rangle \in \mathfrak{I}^U{}_x(Conn)$ oder
$\langle\mathfrak{I}^U{}_x(unit_6), \mathfrak{I}^U{}_x(x)\rangle \in \mathfrak{I}^U{}_x(Conn)$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle U\rangle : \langle UNIT_1, U\rangle \in CONN$ oder $\langle UNIT_6, U\rangle \in CONN$ für $U \in \mathcal{U}^{unit}\} =$
$\{\langle UNIT_2\rangle, \langle UNIT_3\rangle, \langle UNIT_4\rangle, \langle UNIT_5\rangle, \langle UNIT_6\rangle, \langle UNIT_7\rangle, \langle UNIT_8\rangle, \langle UNIT_9\rangle\}$

Frage 5: Welche Dokumente ie_1 und ie_3 stehen in Beziehung miteinander, wenn die Dokumente ie_1 und ie_2 miteinander in Beziehung stehen und die Dokumente ie_2 und ie_3 miteinander in Beziehung stehen (Transitivitätsgesetz) ?

$\mathfrak{I}((\lambda\ x_1\ x_3)\ (\exists\ x_2\ Conn(x_1, x_2) \wedge Conn(x_2, x_3) \rightarrow\ Conn(x_1, x_3)) =$
$\{\langle U_1, U_3\rangle : \mathfrak{I}^{U1\ U3}{}_{x1\ x3}$ ist Modell von $(\exists\ x_2\ Conn(x_1, x_2) \wedge Conn(x_2, x_3) \rightarrow\ Conn(x_1, x_3))$
für $U_1 ,U_3 \in \mathcal{U}^{unit}\} =$
$\{\langle U_1, U_3\rangle : \mathfrak{I}^{U1\ U2\ U3}{}_{x1\ x2\ x3}$ ist Modell von $Conn(x_1, x_2) \wedge Conn(x_2, x_3) \rightarrow\ Conn(x_1, x_3)$
für $U_1 ,U_3 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\} =$
$\{\langle U_1, U_3\rangle : \mathfrak{I}^{U1\ U2\ U3}{}_{x1\ x2\ x3}$ ist Modell von $Conn(x_1, x_3)$ wenn $\mathfrak{I}^{U1\ U2\ U3}{}_{x1\ x2\ x3}$ ist Modell von $Conn(x_1, x_2) \wedge Conn(x_2, x_3)$ für $U_1 ,U_3 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\} =$
$\{\langle U_1, U_3\rangle : \langle\mathfrak{I}^{U1\ U2\ U3}{}_{x1\ x2\ x3}(x_1), \mathfrak{I}^{U1\ U2\ U3}{}_{x1\ x2\ x3}(x_3)\rangle \in \mathfrak{I}^{U1\ U2\ U3}{}_{x1\ x2\ x3}(Conn)$ wenn

$<\mathfrak{I}^{U1\ U2\ U3}_{x1\ x2\ x3}(x_1), \mathfrak{I}^{U1\ U2\ U3}_{x1\ x2\ x3}(x_2)> \in \mathfrak{I}^{U1\ U2\ U3}_{x1\ x2\ x3}(Conn)$ und
$<\mathfrak{I}^{U1\ U2\ U3}_{x1\ x2\ x3}(x_2), \mathfrak{I}^{U1\ U2\ U3}_{x1\ x2\ x3}(x_3)> \in \mathfrak{I}^{U1\ U2\ U3}_{x1\ x2\ x3}(Conn)$
für $U_1, U_3 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\}=$
$\{<U_1, U_3> : < U_1, U_3> \in$ CONN wenn $< U_1, U_2> \in$ CONN und
$< U_2, U_3> \in$ CONN für $U_1, U_3 \in \mathcal{U}^{unit}$ und für mindestens ein $U_2 \in \mathcal{U}^{unit}\}=$
$\{< UNIT_1, UNIT_7>\}$

4.3 LinkQL: Eine Anfragesprache für link-strukturierte Dokumente

Anwender von LinkQL sind Benutzer, die einen Zugriff auf link-strukturierte Dokumente wünschen. LinkQL enthält Sprachelemente für Dokumente und für zweistellige Beziehungen zwischen diesen. Formeln können mit booleschen Operatoren verknüpft werden.
Dokumente werden als einfach (unstrukturiert) vorausgesetzt. Beziehungen zwischen Dokumenten werden durch den Namen ($Conn_1$, $Conn_2$, ...) unterschieden. Beziehungen zwischen Dokumenten sind gerichtet. Die Bestände können Zyklen enthalten.
LinkQL ist eine Erweiterung von SimpleLinkQL. LinkQL enthält Prädikate für Netzwerkfunktionen: Pfad, Vorgänger, Nachfolger, Geschwister, Stationen (vgl. Kap. 2.3 Graphentheorie). Für die Vorgänger/Nachfolgerfunktion wird eine Stufe (Suchtiefe) und für die Netzwerkfunktionen der Namen der Beziehung spezifiziert. Pfade sind elementar, d.h. kein Knoten der Sequenz kommt mehrmals vor. Dadurch wird sichergestellt, daß keine Pfade unendlicher Länge aufgebaut werden.
LinkQL ist eine prädikatenlogische Sprache 1. Stufe. Sie enthält keine Funktionssymbole.

4.3.1 Syntax

Alphabet

1. Individuenkonstanten für Dokumente: unit, $unit_1$, $unit_2$, ...
2. Individuenvariablen für Dokumente: $x, x_1, x_2, ..., x_\alpha, x_\beta, x_\gamma$
3. Individuenkonstanten für eine Stufe: 1, 2, 3, ...
4. Individuenvariable für eine Stufe: s
5. Prädikatenkonstanten (zweistellig): Conn, $Conn_1$, $Conn_2$, ...
6. Prädikatenkonstanten für Netzwerkfunktionen:
   a) Pfad (n-stellig): Path
   b) Nachfolger einer Stufe (zweistellig): $\downarrow$
   c) Vorgänger einer Stufe (zweistellig): $\uparrow$
   d) Nachfolger bis zu einer Stufe (zweistellig): $\Downarrow$
   e) Vorgänger bis zu einer Stufe (zweistellig): $\Uparrow$
   f) Geschwister (zweistellig): $\Leftrightarrow$
   g) Stationen (dreistellig): Θ
7. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation) $\leftrightarrow$ (Äquivalenz)
$\exists$ (es gibt ein), $\forall$ (für alle), λ (alle, die)
8. Technische Symbole: (,)

Formeln

$x_1, ..., x_n, x_\alpha, x_\beta, x_\gamma$ seien Individuenkonstanten oder -variablen für Dokumente, s sei eine Individuenkonstante oder -variable für eine Stufe, Conn sei eine Prädikatenkonstante ($Conn_1$, $Conn_2$, ... s.o.).
1. $Conn(x_\alpha, x_\beta)$ sind Formeln

2. Pfad zwischen x_1 und x_n bzgl. Conn: $Path^{Conn}(x_1, ..., x_n)$,
Nachfolger von x_α der Stufe s bzgl. Conn: $\downarrow^{Conn}_s(x_\alpha, x_\beta)$,
Vorgänger von x_α der Stufe s bzgl. Conn: $\uparrow^{Conn}_s(x_\alpha, x_\beta)$,
Nachfolger von x_α bis zur Stufe s bzgl. Conn: $\Downarrow^{Conn}_s(x_\alpha, x_\beta)$,
Vorgänger von x_α bis zur Stufe s bzgl. Conn: $\Uparrow^{Conn}_s(x_\alpha, x_\beta)$,
Geschwister von x_α bzgl. Conn: $\Leftrightarrow^{Conn}(x_\alpha, x_\beta)$,
Stationen zwischen x_α und x_β bzgl. Conn: $\Theta^{Conn}(x_\alpha, x_\beta, x_\gamma)$ sind Formeln.
Bemerkung: Bei Anwendung der Nachfolger-, Vorgänger-, Geschwister- und Stationenfunktion wird das Ergebnis an der letzten Argumentstelle gezeigt. Z.B. werden die Nachfolger von x_α in x_β gezeigt.
3. Wenn F, F_1 und F_2 Formeln sind, dann auch $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$ $F_1 \leftrightarrow F_2$
$(\exists x_i)$ F, $(\forall x_i)$ F.
4. Das sind alle Formeln.

Anfragen
1. Wenn F eine Formel ist, dann ist $(\lambda\ x_1 \ldots x_n)$ F eine Anfrage.
2. Das sind alle Anfragen.

4.3.2 Semantik

U^{unit} ist eine nichtleere Menge von einfachen Dokumenten.

Interpretation der nichtlogischen Symbole
$\Im(unit_i) \in U^{unit}$, $\Im(x_i) \in U^{unit}$, $\Im(x_\alpha) \in U^{unit}$, $\Im(x_\beta) \in U^{unit}$, $\Im(x_\gamma) \in U^{unit}$,
$\Im(Conn_i) \subseteq U^{unit} \times U^{unit}$, für i, α, β, γ = 1, ..., n (i, α, β, γ können auch weggelassen werden)

Formeln
1. $\Im$ ist Modell von $Conn(x_\alpha, x_\beta)$ gdw $<\Im(x_\alpha), \Im(x_\beta)> \in \Im(Conn)$.
2. $\Im$ ist Modell von $Path^{Conn}(x_1, ..., x_n)$ gdw.
    $\Im$ ist Modell von $Conn(x_1, x_2)$ und $\Im$ ist Modell von $Conn(x_2, x_3)$ und ... und
    $\Im$ ist Modell von $Conn(x_{n-1}, x_n)$ und $\Im(x_1) \neq \Im(x_2) \neq \ldots \neq \Im(x_n)$
  $\Im$ ist Modell von $\downarrow^{Conn}_s(x_\alpha, x_\beta)$ gdw $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_1, x_2, x_3, ..., x_{s-1}, x_\beta)$
  $\Im$ ist Modell von $\uparrow^{Conn}_s(x_\alpha, x_\beta)$ gdw $\Im$ ist Modell von $\downarrow^{Conn}_s(x_\beta, x_\alpha)$
  $\Im$ ist Modell von $\Downarrow^{Conn}_s(x_\alpha, x_\beta)$ gdw
    $\Im$ ist Modell von $\downarrow^{Conn}_s(x_\alpha, x_\beta)$ oder
    $\Im$ ist Modell von $\downarrow^{Conn}_{s-1}(x_\alpha, x_\beta)$ oder ... oder
    $\Im$ ist Modell von $\downarrow^{Conn}_1(x_\alpha, x_\beta)$
  $\Im$ ist Modell von $\Uparrow^{Conn}_s(x_\alpha, x_\beta)$ gdw $\Im$ ist Modell von $\Downarrow^{Conn}_s(x_\beta, x_\alpha)$
  $\Im$ ist Modell von $\Leftrightarrow^{Conn}(x_\alpha, x_\beta)$ gdw
    $\Im$ ist Modell von $Conn(x_\gamma, x_\alpha)$ und $\Im$ ist Modell von $Conn(x_\gamma, x_\beta)$ und $\Im(x_\alpha) \neq \Im(x_\beta)$
  $\Im$ ist Modell von $\Theta^{Conn}(x_\alpha, x_\beta, x_\gamma)$ gdw
    $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_\gamma, x_2, x_3, ..., x_n, x_\beta)$ und
    $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_1, x_\gamma, x_3, ..., x_n, x_\beta)$ und ... und
    $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_1, x_2, x_3, ..., x_\gamma, x_\beta)$ wobei n bel. aber fest
3. $\Im$ ist Modell von $\neg F$ gdw $\Im$ ist nicht Modell von F.
  $\Im$ ist Modell von $F_1 \wedge F_2$ gdw $\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.
  $\Im$ ist Modell von $F_1 \vee F_2$ gdw $\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.
  $\Im$ ist Modell von $F_1 \rightarrow F_2$ gdw $\Im$ ist Modell von F2 wenn $\Im$ ist Modell von F_1.

$\Im$ ist Modell von $F_1 \leftrightarrow F_2$ gdw $\Im$ ist Modell von F1 gdw $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $\exists x_i F$ gdw $\Im^{Ui}_{xi}$ ist Modell von F für mindestens ein $U_i \in U^{unit}$.
$\Im$ ist Modell von $\forall x_i F$ gdw $\Im^{Ui}_{xi}$ ist Modell von F für alle $U_i \in U^{unit}$.

Anfragen
$\Im(\lambda x_1 ... x_n) F) = \{<U_1, ..., U_n>: \Im^{U1 ... U}_{x1 ... xn}$ ist Modell von F für $U_i \in U^{unit}\}$

4.3.3 Beispiele

Es wird der Individuenbereich U^{unit} aus Kapitel 4.2.3 verwendet und um Dokumente erweitert. Es wird eine weitere Relation USED-MATERIAL definiert (in der Graphik s.u. als gestrichelte Linie):

$UNIT_{12}$ = "http://web.nexor.co.uk/aliweb/aliweb"
$UNIT_{13}$ = "http://www.altavista.digital.com"
$UNIT_{14}$ = "http://apollo.co.uk"
$UNIT_{15}$ = "http://www.agentware.com"
$UNIT_{16}$ = "http://www.bigfoot.com"
$UNIT_{17}$ = "http://www.bunyip.com"

$U^{unit} = \{UNIT_1, UNIT_2, UNIT_3, UNIT_4, UNIT_5, UNIT_6, UNIT_7, UNIT_8, UNIT_9, UNIT_{10}, UNIT_{11}, UNIT_{12}, UNIT_{13}, UNIT_{14}, UNIT_{15}, UNIT_{16}, UNIT_{17}\}$

USED-MATERIAL = $\Im$ (Used-Material) = $\{<UNIT_1, UNIT_{12}>, <UNIT_1, UNIT_{13}>, <UNIT_1, UNIT_{14}>, <UNIT_1, UNIT_{15}>, <UNIT_1, UNIT_{16}>, <UNIT_1, UNIT_{17}>\}$

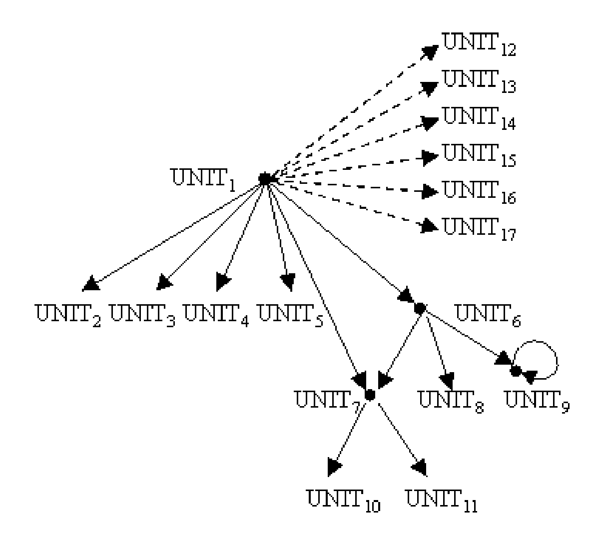


Abbildung 37: Graphische Darstellung der Links

Frage 1: Welche Pfade der Länge 1, 2 oder 3 bestehen zwischen den Dokumenten $unit_1$ und $unit_{11}$ bzgl. der Beziehung Conn ?
Die Frage wird in 3 Teilfragen aufgeteilt dessen Vereinigungsmenge das Gesamtergebnis ist:
$Q1 = (\lambda x_1, x_2, x_3) Path^{Conn}(unit_1, x_1, x_2, x_3, unit_{11})$
$Q2 = (\lambda x_1, x_2) Path^{Conn}(unit_1, x_1, x_2, unit_{11})$
$Q3 = (\lambda x_1) Path^{Conn}(unit_1, x_1, unit_{11}))$
$\Im(Q1) = \{ <U_1, U_2, U_3>: \Im^{U1\ U2\ U3}_{x1\ x2\ x3}$ ist Modell von $Path^{Conn}(unit_1, x_1, x_2, x_3, unit_{11})$

für $U_i \in$ UNIT$\}$ =
$\{<U_1, U_2, U_3>: \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}$ ist Modell von Conn($unit_1, x_1$) und
$\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}$ ist Modell von Conn(x_1, x_2) und
$\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}$ ist Modell von Conn(x_2, x_3) und
$\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}$ ist Modell von Conn($x_3, unit_{11}$)
und $\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(unit_1) \neq \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_1) \neq \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_2) \neq$
$\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_3) \neq \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(unit_{11})$ und für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1, U_2, U_3>: <\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(unit_1), \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_1)> \in \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(Conn)$ und
$< \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_1), \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_2)> \in \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(Conn)$ und
$< \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_2), \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_3)> \in \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(Conn)$ und
$< \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_3), \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(unit_{11})> \in \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(Conn)$
und $\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(unit_1) \neq \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_1) \neq \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_2) \neq$
$\mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(x_3) \neq \mathfrak{I}^{U1\,U2\,U3}{}_{x1\,x2\,x3}(unit_{11})$ und für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1, U_2, U_3>: <UNIT_1, U_1> \in CONN$ und $<U_1, U_2> \in CONN$ und $<U_2, U_3> \in CONN$ und
$<U_3, UNIT_{11}> \in CONN$ und $UNIT_1 \neq U_1 \neq U_2 \neq U_3 \neq UNIT_{11}$ und für $U_i \in \mathit{U}^{unit}\} = \{\}$
$\mathfrak{I}(Q2) = \{ <U_1, U_2>: \mathfrak{I}^{U1\,U2}{}_{x1\,x2}$ ist Modell von $Path^{Conn}(unit_1, x_1, x_2, unit_{11})$ für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1, U_2>: \mathfrak{I}^{U1\,U2}{}_{x1\,x2}$ ist Modell von Conn($unit_1, x_1$) und
$\mathfrak{I}^{U1\,U2}{}_{x1\,x2}$ ist Modell von Conn(x_1, x_2) und
$\mathfrak{I}^{U1\,U2}{}_{x1\,x2}$ ist Modell von Conn($x_2, unit_{11}$) und
$\mathfrak{I}^{U1\,U2}{}_{x1\,x2}(unit_1) \neq \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_1) \neq \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_2) \neq \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(unit_{11})$
und für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1, U_2>: <\mathfrak{I}^{U1\,U2}{}_{x1\,x2}(unit_1), \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_1)> \in \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(Conn)$ und
$<\mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_1), \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_2)> \in \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(Conn)$ und
$<\mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_2), \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(unit_{11})> \in \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(Conn)$ und
$\mathfrak{I}^{U1\,U2}{}_{x1\,x2}(unit_1) \neq \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_1) \neq \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(x_2) \neq \mathfrak{I}^{U1\,U2}{}_{x1\,x2}(unit_{11})$
und für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1, U_2>: <UNIT_1, U_1> \in CONN$ und $<U_1, U_2> \in CONN$ und $<U_2, UNIT_{11}> \in CONN$
und $UNIT_1 \neq U_1 \neq U_2 \neq UNIT_{11}$ und für $U_i \in \mathit{U}^{unit}\}$ =
$\{<UNIT_6, UNIT_7>\}$
$\mathfrak{I}(Q3) = \{ <U_1>: \mathfrak{I}^{U1}{}_{x1}$ ist Modell von $Path^{Conn}(unit_1, x_1, unit_{11})$ für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1>: \mathfrak{I}^{U1}{}_{x1}$ ist Modell von Conn($unit_1, x_1$) und $\mathfrak{I}^{U1}{}_{x1}$ ist Modell von Conn($x_1, unit_{11}$) und
$\mathfrak{I}^{U1}{}_{x1}(unit_1) \neq \mathfrak{I}^{U1}{}_{x1}(x_1) \neq \mathfrak{I}^{U1}{}_{x1}(unit_{11})$ und für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1>: <\mathfrak{I}^{U1}{}_{x1}(unit_1), \mathfrak{I}^{U1}{}_{x1}(x_1)> \in \mathfrak{I}^{U1}{}_{x1}(Conn)$ und
$<\mathfrak{I}^{U1}{}_{x1}(x_1), \mathfrak{I}^{U1}{}_{x1}(unit_{11})> \in \mathfrak{I}^{U1}{}_{x1}(Conn)$ und
$\mathfrak{I}^{U1}{}_{x1}(unit_1) \neq \mathfrak{I}^{U1}{}_{x1}(x_1) \neq \mathfrak{I}^{U1}{}_{x1}(unit_{11})$ und für $U_i \in \mathit{U}^{unit}\}$ =
$\{<U_1>: <UNIT_1, U_1> \in CONN$ und $<U_1, UNIT_{11}> \in CONN$ und
$UNIT_1 \neq U_1 \neq UNIT_{11}$ und für $U_i \in \mathit{U}^{unit}\}$ =
$\{< UNIT_7>\}$
$\mathfrak{I}(Q1) \cup \mathfrak{I}(Q2) \cup \mathfrak{I}(Q3) = \{<UNIT_6, UNIT_7>, <UNIT_7>\}$

Frage 2: Welches sind die Nachfolger der 3. Stufe des Dokument $unit_1$ bzgl. der Beziehung Conn ?
$\mathfrak{I}((\lambda\ x_\beta) \downarrow^{Conn}{}_s(unit_1, x_\beta)$ =
$\{<U_\beta>: \mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\downarrow^{Conn}{}_s(unit_1, x_\beta)$ für $U_\beta \in \mathit{U}^{unit}\}$ =
$\{<U_\beta>: \mathfrak{I}^{U\beta\,U1\,U2}{}_{x\beta\,x1\,x2}$ ist Modell von $Path^{Conn}(unit_1, x_1, x_2, x_\beta)\}$ =
$\{<U_\beta>: \mathfrak{I}^{U\beta\,U1\,U2}{}_{x\beta\,x1\,x2}$ ist Modell von Conn($unit_1, x_1$) und
$\mathfrak{I}^{U\beta\,U1\,U2}{}_{x\beta\,x1\,x2}$ ist Modell von Conn(x_1, x_2) und

$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von Conn(x_2, x_β) und
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(\text{unit}_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_2) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_\beta)$

und für $U_i \in U^{\text{unit}}\} =$
$\{<U_\beta>: <\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(\text{unit}_1), \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_1)> \in \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(\text{Conn})$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_1), \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_2)> \in \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(\text{Conn})$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_2), \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_\beta)> \in \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(\text{Conn})$ und
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(\text{unit}_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_2) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_\beta)$
und für $U_i \in U^{\text{unit}}\} =$
$\{<U_\beta>: \quad <\text{UNIT}_1, U_2> \in \text{CONN}$ und $<U_2, U_3> \in \text{CONN}$ und $<U_3, U_1> \in \text{CONN}$ und
$\text{UNIT}_1 \neq U_\beta \neq U_1 \neq U_2$ und für $U_i \in U^{\text{unit}}\} =$
$\{<\text{UNIT}_{10}>, <\text{UNIT}_{11}>\}$

Frage 3: Welches sind die Vorgänger der 2. Stufe des Dokuments unit_{10} bzgl. der Beziehung Conn ?
$\mathfrak{I}((\lambda\ x_\alpha) \uparrow^{\text{Conn}}{}_2(x_\alpha, \text{unit}_{10})) =$
$\{<U_\alpha>: \mathfrak{I}^{U\alpha}{}_{x\alpha}$ ist Modell von $\uparrow^{\text{Conn}}{}_2(x_\alpha, \text{unit}_{10})$ für $U_i \in U^{\text{unit}}\} =$
$\{<U_\alpha>: \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}$ ist Modell von $\text{Path}^{\text{Conn}}(x_\alpha, x_1, \text{unit}_{10})$ und
$\mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_\alpha) \neq \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_1) \neq \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(\text{unit}_{10})$ und für $U_i \in U^{\text{unit}}\} =$
$\{<U_\alpha>: \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}$ ist Modell von Conn(x_α, x_1) und
$\mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}$ ist Modell von Conn(x_1, unit_{10}) und $\mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_1) \neq \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_\alpha)$
und für $U_i \in U^{\text{unit}}\} =$
$\{<U_\alpha>: <\mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_\alpha), \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_1)> \in \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(\text{Conn})$ und
$<\mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_1), \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(\text{unit}_{10})> \in \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(\text{Conn})$
und $\mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_1) \neq \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(x_\alpha) \neq \mathfrak{I}^{U\alpha\ U1}{}_{x\alpha\ x1}(\text{unit}_{10})$ und für $U_i \in U^{\text{unit}}\} =$
$\{<U_\alpha>: <U_\alpha, U_1> \in \text{CONN}$ und $< U_1, \text{UNIT}_{10}> \in \text{CONN}$ und $U_\alpha \neq U_1 \neq \text{UNIT}_{10}$
und für $U_i \in U^{\text{unit}}\} =$
$\{<\text{UNIT}_1>, <\text{UNIT}_6>\}$

Frage 4: Welches sind die Geschwister des Dokuments unit_7 bzgl. der Beziehung Conn ?
$\mathfrak{I}((\lambda\ x_1) \Leftrightarrow^{\text{Conn}}(\text{unit}_7, x_1)) =$
$\{<U_1>: \mathfrak{I}^{U1}{}_{x1}$ ist Modell von $\Leftrightarrow^{\text{Conn}}(\text{unit}_7, x_1)$ für $U_i \in U^{\text{unit}}\} =$
$\{<U_1>: \mathfrak{I}^{U1\ U2}{}_{x1\ x2}$ ist Modell von Conn(x_2, unit_7) und
$\mathfrak{I}^{U1\ U2}{}_{x1\ x2}$ ist Modell von Conn(x_2, x_1) und
$\mathfrak{I}^{U1\ U2}{}_{x1\ x2}(\text{unit}_7) \neq \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(x_1)$ und für $U_i \in U^{\text{unit}}\} =$
$\{<U_1>: <\mathfrak{I}^{U1\ U2}{}_{x1\ x2}(x_2), \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(\text{unit}_7)> \in \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(\text{Conn})$ und
$<\mathfrak{I}^{U1\ U2}{}_{x1\ x2}(x_2), \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(x_1)> \in \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(\text{Conn})$ und
$\mathfrak{I}^{U1\ U2}{}_{x1\ x2}(\text{unit}_7) \neq \mathfrak{I}^{U1\ U2}{}_{x1\ x2}(x_1)$ und für $U_i \in U^{\text{unit}}\} =$
$\{<U_1>: <U_2, \text{UNIT}_7> \in \text{CONN}$ und $<U_2, U_1> \in \text{CONN}$ und
$\text{UNIT}_7 \neq U_2$ und für $U_i \in U^{\text{unit}}\} =$
$\{<\text{UNIT}_2>, <\text{UNIT}_3>, <\text{UNIT}_4>, <\text{UNIT}_5>, <\text{UNIT}_6>, <\text{UNIT}_7>, <\text{UNIT}_8>, <\text{UNIT}_9>\}$

Frage 5: Welche Stationen liegen zwischen den Dokumenten unit_1 und unit_{11} bzgl. der Beziehung Conn (bei Pfaden höchstens der Länge 2)?
$\mathfrak{I}((\lambda\ x_\gamma)\ \Theta^{\text{Conn}}(\text{unit}_1, \text{unit}_{10}, x_\gamma)) =$
$\{<U_\gamma>: \mathfrak{I}^{U\gamma}{}_{x\gamma}$ ist Modell von $\Theta(\text{unit}_1, \text{unit}_{10}, x_\gamma, \text{Conn})$ für $U_i \in U^{\text{unit}}\} =$
$\{<U_\gamma>: \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x1}$ ist Modell von $\text{Path}^{\text{Conn}}(\text{unit}_1, x_\gamma, x_2, \text{unit}_{10})$ oder

$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}$ ist Modell von $Path^{Conn}(unit_1, x_1, x_\gamma, unit_{10})$
für $n = 2$ und für $U_i \in U^{unit}\}$
$\{<U_\gamma>$: $\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}$ ist Modell von $Conn(unit_1, x_\gamma)$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}$ ist Modell von $Conn(x_\gamma, x_2)$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}$ ist Modell von $Conn(x_2, unit_{10})$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_1) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_2) \neq$
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_{10})$ oder
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}$ ist Modell von $Conn(unit_1, x_1)$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}$ ist Modell von $Conn(x_1, x_\gamma)$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}$ ist Modell von $Conn(x_\gamma, unit_{10})$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_1) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_1) \neq$
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_{10})$ und für $U_i \in U^{unit}\} =$
$\{<U_\gamma>$: $<\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_1), \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma)> \in \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(Conn)$ und
$<\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma), \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_3)> \in \mathfrak{I}\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(Conn)$ und
$<\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_3), \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_{10})> \in \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(Conn)$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_1) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_2) \neq$
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_{10})$ oder
$<\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_1), \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_2)> \in \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(Conn)$ und
$<\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_2), \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma)> \in \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(Conn)$ und
$<\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma), \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_{10})> \in \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(Conn)$ und
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(unit_1) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_\gamma) \neq \mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}(x_1) \neq$
$\mathfrak{I}^{U\gamma\ U1\ U2}{}_{x\gamma\ x1\ x2}\ (unit_{10})$ und für $U_i \in U^{unit}\} =$
$\{<U_\gamma>$: $<UNIT_1, U_\gamma> \in CONN$ und $<U_\gamma, U_2> \in CONN$ und $< U_2, UNIT_{10}> \in CONN$
und $UNIT_1 \neq U_\gamma \neq U_2 \neq UNIT_{10}$ oder
$<UNIT_1, U_1> \in CONN$ und $<U_1, U_\gamma> \in CONN$ und $< U_\gamma, UNIT_{10}> \in CONN$ und
und $UNIT_1 \neq U_\gamma \neq U_2 \neq UNIT_{10}$ und für $U_i \in U^{unit}\} =$
$\{<UNIT_6>, <UNIT_7>\}$

Frage 6: Welches sind die Nachfolger bis zur 3. Stufe des Dokuments $unit_1$ bzgl. der Beziehungen Conn oder Used-Material ?
$\mathfrak{I}((\lambda\ x_\beta) \Downarrow^{Conn}{}_3(unit_1, x_\beta) \vee \Downarrow^{Used\text{-}Material}{}_3(unit_1, x_\beta)) =$
$\{<U_\beta>$: $\mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\Downarrow^{Conn}{}_3(unit_1, x_\beta) \vee \Downarrow^{Used\text{-}Material}{}_3(unit_1, x_\beta)$ für $U_i \in U^{unit}\} =$
$\{<U_\beta>$: $\mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\downarrow^{Conn}{}_3(unit_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\downarrow^{Conn}{}_2(unit_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\downarrow^{Conn}{}_1(unit_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\downarrow^{Used\text{-}Material}{}_3(unit_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\downarrow^{Used\text{-}Material}{}_2(unit_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta}{}_{x\beta}$ ist Modell von $\downarrow^{Used\text{-}Material}{}_1(unit_1, x_\beta)$ für $U_i \in U^{unit}\} =$
$\{<U_\beta>$: $\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von $Path^{Conn}(unit_1, x_1, x_2, x_\beta)$ oder
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von $Path^{Conn}(unit_1, x_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von $Path^{Conn}(unit_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von $Path^{Used\text{-}Material}(unit_1, x_1, x_2, x_\beta)$ oder
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von $Path^{Used\text{-}Material}(unit_1, x_1, x_\beta)$ oder
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von $Path^{Used\text{-}Material}(unit_1, x_\beta)$ und
$\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(unit_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_2) \neq \mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}(x_\beta)$
und für $U_i \in U^{unit}\} =$
$\{<U_\beta>$: $\mathfrak{I}^{U\beta\ U1\ U2}{}_{x\beta\ x1\ x2}$ ist Modell von $Conn(unit_1, x_1)$ und

$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Conn(x_1, x_2) und
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Conn(x_2, x_β) oder
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Conn($unit_1$, x_1) und
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Conn(x_1, x_β) oder
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Conn($unit_1$, x_β) oder
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Used-Material($unit_1$, x_1) und
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Used-Material(x_1, x_2) und
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Used-Material(x_2, x_β) oder
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Used-Material($unit_1$, x_1) und
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Used-Material(x_1, x_β) oder
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}$ ist Modell von Used-Material($unit_1$, x_β) und
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_2) \neq \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)$
und für $U_i \in \mathcal{U}^{unit}$} =

{$<U_\beta>$: $<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Conn)$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_2)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Conn)$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_2), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Conn)$ oder
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Conn)$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Conn)$ oder
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Conn)$ oder
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Used\text{-}Material)$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_2)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Used\text{-}Material)$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_2), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Used\text{-}Material)$ oder
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Used\text{-}Material)$ und
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Used\text{-}Material)$ oder
$<\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1), \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)> \in \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(Used\text{-}Material)$ und
$\mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(unit_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_1) \neq \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_2) \neq \mathfrak{I}^{U\beta\ U1\ U2}_{x\beta\ x1\ x2}(x_\beta)$
und für $U_i \in \mathcal{U}^{unit}$}=

{$<U_\beta>$: $<UNIT_1, U_1> \in CONN$ und $<U_1, U_2> \in CONN$ und $<U_2, U_\beta> \in CONN$ oder
$<UNIT_1, U_1> \in CONN$ und $<U_1, U_\beta> \in CONN$ oder
$<UNIT_1), U_\beta> \in CONN$ oder
$<UNIT_1, U_1> \in$ USED-MATERIAL und $<U_1, U_2> \in$ USED-MATERIAL und
$<U_2, U_\beta> \in$ USED-MATERIAL oder
$<UNIT_1, U_1> \in$ USED-MATERIAL und $<U_1, U_\beta> \in$ USED-MATERIAL oder
$<UNIT_1), U_\beta> \in$ USED-MATERIAL und
$UNIT_1 \neq U_\beta \neq U_1 \neq U_2$ und für $U_i \in \mathcal{U}^{unit}$}=

{$<UNIT_2>$, $<UNIT_3>$, $<UNIT_4>$, $<UNIT_5>$, $<UNIT_6>$, $<UNIT_7>$, $<UNIT_8>$, $<UNIT_9>$, $<UNIT_{10}>$, $<UNIT_{11}>$, $<UNIT_{12}>$, $<UNIT_{13}>$, $<UNIT_{14}>$, $<UNIT_{15}>$, $<UNIT_{16}>$, $<UNIT_{17}>$}

4.4 Nested&LinkQL: Eine Anfragesprache für geschachtelte und link-strukturierte Dokumente

Anwender von Nested&LinkQL sind Benutzer, die einen Zugriff auf geschachtelte und link-strukturierte Dokumente wünschen.
In Nested&LinkQL haben Individuen und Funktionen eine bestimmte Sorte, Beziehungen bestehen zwischen Elementen unterschiedlicher Sorte. Geschachtelte Dokumente werden mit den Funktionen aufgebaut (vgl. Abiteboul, Beeri (1995)). Jedem Funktionssymbol wird genau

eine Abbildung zugeordnet. Dadurch können Sammlungen von Dokumenten wie zum Beispiel Array, Menge, Liste und Record mit Nested&LinkQL nachgebildet werden.

In objektorientierten Sprachen ist die Bildung von Objekten im Gegensatz zu der hier vorgestellten Sprache keine Funktion im mathematischen Sinne. Die Anwendung eines Konstruktors kann mehrere Werte liefern. Beispielsweise liefert STRING("Hallo") zweimalig angewendet zwei unterschiedliche Objekte (mit verschiedenen Identifikationskennzeichen).
Für die Bildung von Teilobjekten könnte als Alternative zu den mehrstelligen Funktionssymbolen der Punktoperator verwendet werden. Beispielsweise würde dann mit html.body.ul eine ungeordnete Liste bezeichnet werden. Allerdings hätte dies den Nachteil, daß mehrere gleichartige Teilobjekte derselben Ebene nicht mehr exakt bestimmt werden könnten. Beispielsweise wäre html.body.h1 in dem Dokument $UNIT_1$ in Kapitel 4.4.3 mehrdeutig.
Als Anwendungsgebiet werden Internet-Informationssysteme gewählt, die Dokumente überwiegend mit Teilsprachen von SGML (vgl. ISO 8879), wie z.B. XML strukturieren. Dementsprechend werden in Nested&LinkQL Funktionskonstanten für SGML-Elemente bereitgestellt (z.B. html, xml, book, ul, li, usw.).
Nested&LinkQL enthält Prädikate, mit denen Teile, Behälter und Geschwister von Dokumenten bzw. Teilen von Dokumenten bestimmt werden können. Für die Teile- und Behälterprädikate kann eine Stufe (Suchtiefe) spezifiziert werden. Mit HyQ (vgl. DeRose, Durand (1994) S. 165) und XQL (Robie, Lapp, Schach (1998)) ist eine Anfrage nach Teilen von Dokumenten ebenfalls möglich.
Mit dem Prädikatensymbol Conn werden zweistellige Beziehungen zwischen Dokumenten bzw. zwischen Teilen von Dokumenten gebildet (bei Conklin (1987) S. 34 Regionen genannt). Dokumente können überlappen.
Mit Prädikatenkonstanten zweiter Stufe werden Eigenschaften von Beziehungen ausgedrückt. Wenn nicht anders angegeben, haben Prädikate in Nested&LinkQL jedoch die erste Stufe.
Nested&LinkQL ist eine prädikatenlogische Sprache 2. Stufe. Sie enthält Quantoren (Existenz- und Allquantor) für Funktionen und Prädikate, den Lambda-Operator für Prädikate und Prädikatenkonstanten der zweiten Stufe. Wenn Nested&LinkQL ohne diese aufgebaut würde, wäre Nested&LinkQL eine mehrsortige, prädikatenlogische Sprache 1. Stufe.

4.4.1 Syntax

Sorten:

1. string, int, bool, unit, html, head, body, title, chapter, abstract, ... sind Sorten.
2. Das sind alle Sorten.

Im Folgenden benutzen wir die Variablen s, s_1, ..., s_n für diese Sorten.

Alphabet:

1. Individuenkonstanten: string, $string_1$, ... , int, int_1, ..., true, false, unit, $unit_1$, ..., html, $html_1$, ..., head, $head_1$, ... body, $body_1$, ..., title, $title_1$, ..., chapter, $chapter_1$, ... abstract, $abstract_1$, ..., ...
2. Individuenvariablen verschiedener Sorten: x^{string}, $x^{string}{}_1$, ... , x^{int}, $x^{int}{}_1$, ... , x^{bool}, $x^{bool}{}_1$, ... , x^{unit}, $x^{unit}{}_1$, ... , x^{html}, $x^{html}{}_1$, ... , x^{head}, $x^{head}{}_1$, ... , x^{body}, $x^{body}{}_1$, ... , x^{title}, $x^{title}{}_1$, ..., $x^{chapter}$, $x^{chapter}{}_1$, ..., $x^{abstract}$, $x^{abstract}{}_1$, ..., x, x_1, ..., ...
3. Individuenkonstanten für Stufen: 1, 2, ...
4. Individuenvariable für eine Stufe: m
5. Funktionskonstanten von Sorten s_1, ..., s_n auf die angegebene Sorte: $html^{s:html}$, $head^{s1,...,sn:head}$, $body^{s1,...,sn:body}$, $title^{s1,...,sn:title}$, $chapter^{s1,...,sn:chapter}$, $abstract^{s1,...,sn:abstract}$, ...
6. Funktionsvariablen von Sorten s_1, ..., s_n auf die Sorte s: $f^{s1,...,sn:s}{}_1$, $f^{s1,...,sn:s}{}_2$, ...
7. Prädikatenkonstanten

a) Dokumente (einstellig): $Unit^{s}$
b) Beziehungen (zweistellig): $Conn^{s1, s2}$
c) Gleichheit (zweistellig): $Equal^{s1, s2}$
8. Prädikatenkonstanten für Teile:
a) Teile einer Stufe (zweistellig): $\triangledown$
b) Teile bis zu einer Stufe (zweistellig): $\blacktriangledown$
c) Behälter einer Stufe (zweistellig): $\triangle$
d) Behälter bis zu einer Stufe (zweistellig): $\blacktriangle$
e) Geschwister (zweistellig): $\diamond$
9. Prädikatenvariablen: ${P^{s1, s2}}_1$, ${P^{s1, s2}}_2$, ...
10. Prädikatenkonstanten 2. Stufe: Reflexive, Nonreflexive, Irreflexive, Symmetric, Asymmetric, Antisymmetric, Transitive, NegativelyTransitive, Nontransitive, EquivalenceRelation, Complete, StronglyComplete, ...
11. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation), $\leftrightarrow$ (Äquivalenz), $\exists$ (es gibt ein), $\forall$ (für alle), λ (die Menge)
12. Technische Symbole: (,)

Mit diesen Symbolen werden induktiv die Terme und Formeln gebildet:

Terme:
1. Individuenkonstanten und -variablen der Sorte s sind Terme der Sorte s.
2. Wenn t_1 Term der Sorte s_1 , t_2 Term der Sorte s_2 ,... , t_n Term der Sorte s_n ist, und f ein Funktionssymbol (Konstante oder Variable) von Sorten s_1 ,... , s_n auf s ist, dann ist
$f^{s1, \ldots, sn:s}(t_1, t_2, \ldots , t_n)$ Term der Sorte s.
3. Das sind alle Terme.

Formeln:
t sei Term der Sorte s, t_1 sei Term der Sorte s_1 , t_2 sei Term der Sorte s_2 ,... , t_n sei Term der Sorte s_n, m sei eine Stufe (Konstante oder Variable).

1. $Unit^{s}(t)$, $Conn^{s1, s2}(t_1, t_2)$, $Equal^{s1, s2}(t_1, t_2)$ sind Formeln.
$P^{s1, s2}(t_1, t_2)$ ist eine Formel.
2. Teile von t_1 der Stufe m: $\triangledown_m(t, x)$
Teile von t_1 bis zur Stufe m: $\blacktriangledown_m(t, x)$
Behälter von t_1 der Stufe m: $\triangle_m(t, x)$
Behälter von t_1 bis zur Stufe m: $\blacktriangle_m(t, x)$
Geschwister von t_1: $\diamond(t_1, x)$ sind Formeln.
Bemerkung: Bei Anwendung der Teil-, Behälter- und Geschwisterfunktion zeigen wir das Ergebnis jeweils an der zweiten Argumentstelle x.
3. Wenn F, F_1 und F_2 Formeln sind, dann sind auch $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$, $\exists x^s F$, $\forall x^s F$, $\exists P^{s1,s2} F$ und $\forall P^{s1,s2} F$ Formeln.
4. Wenn $P^{s1, s2}$ ein Prädikatensymbol (Konstante oder Variable) der Sorten s_1, s_2 ist und Prop eine Prädikatenkonstante zweiter Stufe für Eigenschaften von Beziehungen ist, dann ist auch $Prop(P^{s1, s2})$ eine Formel.
5. Das sind alle Formeln.

Anfragen:

1. Wenn F eine Formel ist und $x^{s1}_1 \dots x^{sn}_n$ Individuenvariablen der Sorten $s_1, \dots, s_n$ sind, dann ist $(\lambda\ x^{s1}_1 \dots x^{sn}_n)$ F eine Anfrage.
2. Wenn F eine Formel ist, dann ist $(\lambda\ P^{s1,\ s2})$ F eine Anfrage.
3. Das sind alle Anfragen.

4.4.2 Semantik

In Dokumentenbeschreibungssprachen wie SGML werden geschachtelte Dokumente durch sogenannte "Elemente" aufgebaut. Elemente sind beispielsweise geordnete und ungeordnete Listen („OL“ und „UL“) von Listeneinträgen („LI“), Kapitel („CHAPTER“) und Überschriften („HEADING“). Dafür werden nichtleere Individuenbereiche (Elementmengen) unterschiedlicher Sorte zugrundegelegt: U^{html}, U^{head}, U^{body}, U^{title}, $U^{chapter}$, $U^{abstract}$, $U^{heading}$, U^{ol}, U^{ul}, U^{li}, U^{string}, U^{int}, U^{bool}, ... Für Variablen bel. Sorte wird die Vereinigungsmenge aller nichtleeren Individuenbereiche U zugrundegelegt. Für Funktionen werden Abbildungen und für Prädikate Relationen zugrundegelegt. Für das Prädikat Equal wird die Gleichheitsrelation zugrundegelegt.
Ein Teil eines Dokuments kann Bestandteil mehrerer Dokumente sein. In diesem Fall reichen Bäume als Darstellungsmittel nicht mehr aus und in diesem Fall werden gerichtete azyklische Graphen verwendet. Beispielsweise könnte die ungeordnete Liste aus dem Beispiel (siehe Kapitel 4.4.3) Teil eines anderen Dokuments sein:

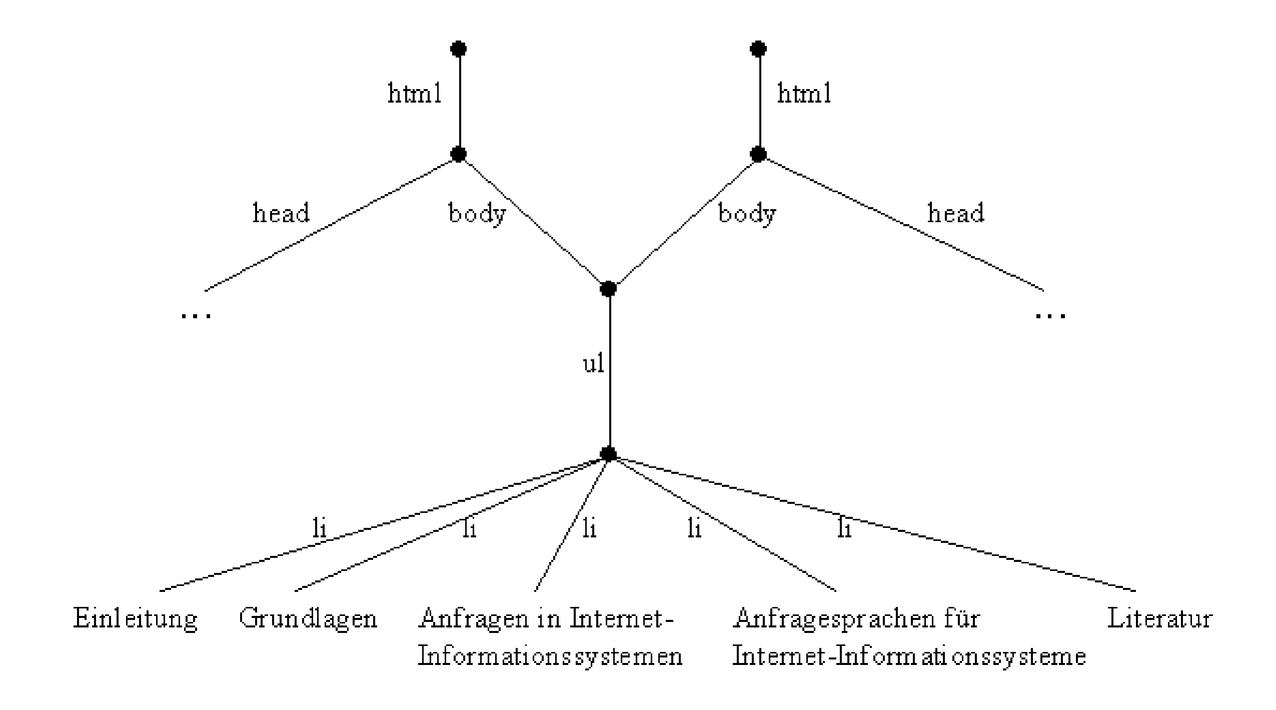


Abbildung 38: Ein Teildokument als Teil mehrerer Dokumente

Links können sowohl zwischen Dokumenten als auch zwischen Teilen von Dokumenten gebildet werden.

Interpretation der Individuen-, Funktions- und Prädikatensymbole

1. $\Im(string_i) \in U^{string}$, $\Im(int_i) \in U^{int}$, $\Im(true) \in U^{bool}$, $\Im(false) \in U^{bool}$, $\Im(html_i) \in U^{html}$, $\Im(head_i) \in U^{head}$, $\Im(body_i) \in U^{body}$, $\Im(title_i) \in U^{title}$, $\Im(chapter_i) \in U^{chapter}$, $\Im(abstract_i) \in U^{abstract}$, ... für i = 1, ..., n (i kann auch weggelassen werden)

$\Im(x^{string}_i) \in U^{string}$, $\Im(x^{int}_i) \in U^{int}$, $\Im(x^{bool}_i) \in U^{bool}$, $\Im(x^{html}_i) \in U^{html}$, $\Im(x^{head}_i) \in U^{head}$, $\Im(x^{body}_i) \in U^{body}$, $\Im(x^{title}_i) \in U^{title}$, $\Im(x^{chapter}_i) \in U^{chapter}$, $\Im(x^{abstract}_i) \in U^{abstract}$, ..., $\Im(x_i) \in U$
für i = 1, ..., n (i kann auch weggelassen werden)
2. $\Im(html^{s:html}) \in \{U^s \to_{HTML} U^{html}\}$, $\Im(head^{s1,\ \ldots,\ sn:head}) \in \{U^{s1} \times \ldots \times U^{sn} \to_{HEAD} U^{head}\}$, $\Im(body^{s1,\ \ldots,\ sn:body}) \in \{U^{s1} \times \ldots \times U^{sn} \to_{BODY} U^{body}\}$, $\Im(title^{s1,\ \ldots,sn:title}) \in \{U^{s1} \times \ldots \times U^{sn} \to_{TITLE} U^{title}\}$, $\Im(chapter^{s1,\ \ldots,sn:chapter}) \in \{U^{s1} \times \ldots \times U^{sn} \to_{CHAPTER} U^{chapter}\}$, $\Im(abstract^{s1,\ \ldots,sn:abstract}) \in \{U^{s1} \times \ldots \times U^{sn} \to_{ABSTRACT} U^{abstract}\}$, ...
$\Im(f^{s1,\ \ldots\ ,\ sn:sm}_i) \in \{U^{s1} \times \ldots \times U^{sn} \to \Im(s_m)\}$, für i = 1, ..., n
3. $\Im(UNIT^s) \subseteq \Im(s)$, $\Im(Conn^{s1,\ s2}) \subseteq U^{s1} \times U^{s2}$, $\Im(Equal) \subseteq U \times U$,
$\Im(P^{s1,\ s2}_i) \subseteq U^{s1} \times U^{s2}$,

Interpretation der Terme
k sei Funktionskonstante
4. $\Im(k^{s1,\ \ldots\ ,\ sn:s}_i(t_1, \ldots , t_n)) = \Im(k^{s1,\ \ldots\ ,\ sn:s}_i)\ (\Im(t_1), \Im(t_2), \ldots , \Im(t_n))$,
$\Im(f^{s1,\ \ldots\ ,\ sn:s}_i(t_1, \ldots , t_n)) = \Im(f^{s1,\ \ldots\ ,\ sn:s}_i)\ (\Im(t_1), \Im(t_2), \ldots , \Im(t_n))$,

Interpretation der Formeln
5. $\Im$ ist Modell von $Unit^s(t)$ gdw $<\Im(t)> \in \Im(Unit^s)$,
$\Im$ ist Modell von $Conn^{s1,\ s2}(t_1, t_2)$ gdw $<\Im(t_1), \Im(t_2)> \in \Im(Conn^{s1,\ s2})$,
$\Im$ ist Modell von $Equal(t_1, t_2)$ gdw $\Im(t_1) = \Im(t_2)$,
$\Im$ ist Modell von $P^{s1,\ s2}(t_1, t_2)$ gdw $<\Im(t_1), \Im(t_2)> \in \Im(P^{s1,\ s2})$,
6. f sei Funktion (Konstante oder Variable) bel. Sorte und bel. aber fester Stellenzahl
$\Im$ ist Modell von $\triangledown_m(t, x)$ gdw
$\Im$ ist Modell von $(\exists f, \exists x_1, \ldots, \exists x_n)\ \triangledown_{m-1}(t, f(x_1, \ldots, x, \ldots , x_n))$
$\Im$ ist Modell von $\triangledown_1(t, x)$ gdw
$\Im$ ist Modell von $(\exists f, \exists x_1, \ldots, \exists x_n)\ Equal(t, f(x_1, \ldots, x, \ldots , x_n))$
$\Im$ ist Modell von $\blacktriangledown_m(t, x)$ gdw
$\Im$ ist Modell von $\triangledown_m(t, x)$ und $\Im$ ist Modell von $\triangledown_{m-1}(t, x)$ und ... und
$\Im$ ist Modell von $\triangledown_1(t, x)$
$\Im$ ist Modell von $\triangle_m(t, x)$ gdw
$\Im$ ist Modell von $(\exists f, \exists x_1, \ldots, \exists x_n)\ \triangle_{m-1}(f(x_1, \ldots, t, \ldots , x_n), x)$
$\Im$ ist Modell von $\triangle_1(t, x)$ gdw
$\Im$ ist Modell von $(\exists f, \exists x_1, \ldots, \exists x_n)\ Equal(x, f(x_1, \ldots, t, \ldots , x_n))$
$\Im$ ist Modell von $\blacktriangle_m(t, x)$ gdw
$\Im$ ist Modell von $\triangle_m(t, x)$ und $\Im$ ist Modell von $\triangle_{m-1}(t, x)$ und ... und
$\Im$ ist Modell von $\triangle_1(t, x)$
$\Im$ ist Modell von $\diamond(t, x)$ gdw
$\Im$ ist Modell von $\exists f, \exists x_1\ Equal(f(t, x, \ldots, x), x_1) \wedge Equal(f(x, t, \ldots, x), x_1) \wedge \ldots \wedge$
$Equal(f(x, x, \ldots, t), x_1)$
6. $\Im$ ist Modell von $\neg F$ gdw $\Im$ ist nicht Modell von F.
$\Im$ ist Modell von $(F_1 \wedge F_2)$ gdw $\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $(F_1 \vee F_2)$ gdw $\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $(F_1 \to F_2)$ gdw $\Im$ ist Modell von F2 wenn $\Im$ ist Modell von F_1.
$\Im$ ist Modell von $(F_1 \leftrightarrow F_2)$ gdw ($\Im$ ist Modell von F1 gdw $\Im$ ist Modell von F_2).
$\Im$ ist Modell von $(\exists\ x^s_i)$ F gdw $\Im^U_{xi}$ ist Modell von F für mindestens ein $U \in U^s$.
$\Im$ ist Modell von $(\forall\ x^s_i)$ F gdw $\Im^U_{xi}$ ist Modell von F für alle $U \in U^s$.
$\Im$ ist Modell von $(\exists\ f^{s1,\ \ldots\ ,\ sn:s})$ F gdw

$\Im^{FU}{}_f$ ist Modell von F für mindestens ein FU $\in \{U^{s1} \times ... \times U^{sn} \rightarrow U^{s}\}$.
$\Im$ ist Modell von $(\forall\ f^{s1, ..., sn:s})$ F gdw
$\Im^{FU}{}_f$ ist Modell von F für alle FU $\in \{U^{s1} \times ... \times U^{sn} \rightarrow U^{s}\}$.
$\Im$ ist Modell von $(\exists\ f)$ F gdw $\Im^{FU}{}_f$ ist Modell von F für mindestens ein FU $\in \{ \rightarrow \}$.
$\Im$ ist Modell von $(\forall\ f)$ F gdw $\Im^{FU}{}_f$ ist Modell von F für alle FU $\in \{ \rightarrow \}$.
$\Im$ ist Modell von $(\exists\ P^{s1, s2})$ F gdw
$\Im^{rel}{}_P$ ist Modell von F für mindestens eine Relation über $U^{s1} \times U^{s2}$.
$\Im$ ist Modell von $(\forall\ P^{s1, s2})$ F gdw $\Im^{rel}{}_P$ ist Modell von F für alle Relationen über $U^{s1} \times U^{s2}$.
$\Im$ ist Modell von $Prop(P^{s1, s2})$ gdw $<\Im(P^{s1, s2})> \in \Im(Prop)$
Prop sei Prädikatenkonstante 2. Stufe

Interpretation der Anfragen
7. $\Im((\lambda\ x^{s1}{}_1 ... x^{sn}{}_n)\ F) =$
$\{<U_1, ..., U_n>: \Im^{U1...Un}{}_{x1...xn}$ ist Modell von F für $U_1 \in U^{s1}, ..., U_n \in U^{sn}\}$.
$\Im((\lambda\ P^{s1, s2})\ F) = \{<rel>: \Im^{rel}{}_P$ ist Modell von F$\}$.

4.4.3 Beispiele

Für die Beispielanfragen werden Individuenbereiche, Abbildungen und Relationen vorgegeben. Zunächst wird das HTML-Dokument diss.html als Beispiel für geschachtelte Dokumente vorgestellt. Individuenbereiche sind die Grundlage für die Abbildungen. Mit den Abbildungen wird die Struktur von 6 Dokumenten ($HTML_1$ – $HTML_6$) dargestellt. Mit der einstelligen Relation UNIT werden Dokumente ausgezeichnet, mit der zweistelligen Relation CONN werden Beziehungen zwischen den Dokumenten hergestellt.

```
<HTML>
<BASE HREF="http://www.cs.tu-berlin.de/~josefw/phd/index.html">
<HEAD>
<TITLE>Anfragesprachen für Internet-Informationssysteme</TITLE>
</HEAD>
<BODY>
<H1>Anfragesprachen für Internet-Informationssysteme</H1>
<H1>Inhalt</H1>
<UL>
<LI><A HREF="http://www.cs.tu-berlin.de/~josefw/phd/introduction.html">Einleitung</A>
<LI><A HREF="http://www.cs.tu-berlin.de/~josefw/phd/state.html">Grundlagen</A>
<LI><A HREF="http://www.cs.tu-berlin.de/~josefw/phd/query.html">Anfragen in Internet-
Informationssystemen</A>,
<LI><A HREF="http://www.cs.tu-berlin.de/~josefw/phd/hynternetQL.html">
Anfragesprachen für Internet-Informationssysteme</A>
<LI><A HREF="http://www.cs.tu-berlin.de/~josefw/phd/literature.html">Literatur</A>
</UL>
<IMG SRC="http://www.cs.tu-berlin.de/~josefw/joe.gif">
<A HREF="http://www.cs.tu-berlin.de/~josefw/phd/joe.mpg">Videodarstellung</A>
</BODY>
</HTML>
```

Abbildung 39: Das HTML-Dokument diss.html

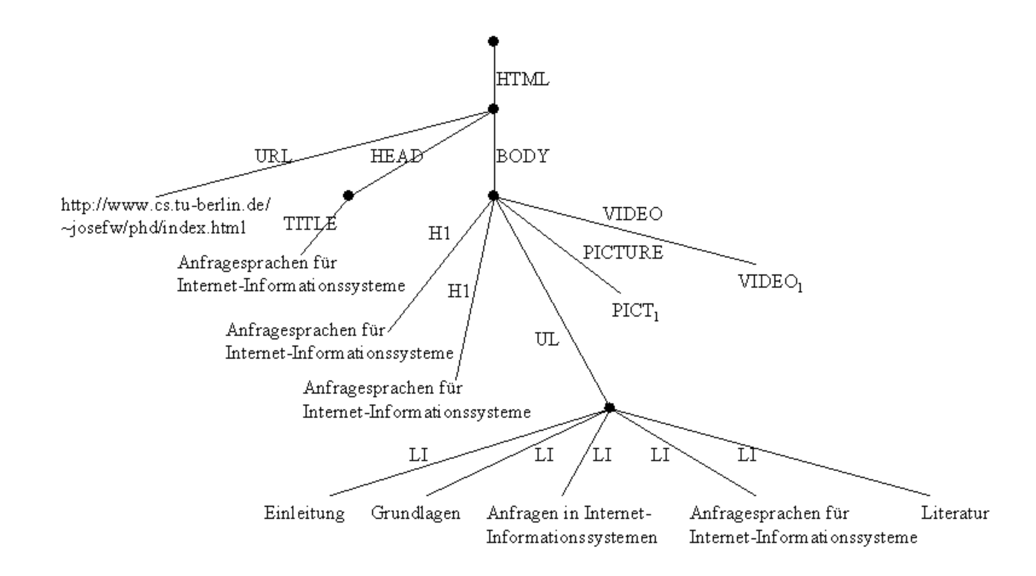


Abbildung 40: Funktionale Baumstruktur von diss.html

Individuenbereiche:

U^{string} = {"Abiteboul, Beeri (1995) ...", "Anfragen in Internet-Informationssystemen", "Anfragesprachen für Internet-Informationssysteme", "Das Internet entstand Anfang der 70er Jahre als ...", "Einleitung", "Grundlagen", "In den Internet-Informationssystemen stehen ...", "Literatur", "Seit Entstehung des World Wide Web (WWW) hat sich ...", "Wir behandeln die Entwicklung von Anfragesprachen ..."}

$U^{picture} = \{PICT_1\}$

$U^{video} = \{VIDEO_1\}$

$U^{url} = \{URL_1, URL_2, URL_3, URL_4, URL_5, URL_6\}$

$U^{title} = \{TITLE_1, TITLE_2, TITLE_3, TITLE_4, TITLE_5, TITLE_6\}$

$U^{h1} = \{H1_1, H1_2, H1_3, H1_4, H1_5, H1_6, H1_7\}$

$U^{li} = \{LI_1, LI_2, LI_3, LI_4, LI_5\}$

$U^{ul} = \{UL_1\}$

$U^{head} = \{HEAD_1, HEAD_2, HEAD_3, HEAD_4, HEAD_5, HEAD_6\}$

$U^{body} = \{ BODY_1, BODY_2, BODY_3, BODY_4, BODY_5, BODY_6\}$

$U^{html} = \{HTML_1, HTML_2, HTML_3, HTML_4, HTML_5, HTML_6\}$

$U = U^{string} \cup U^{picture} \cup U^{video} \cup U^{url} \cup U^{title} \cup U^{h1} \cup U^{li} \cup U^{ul} \cup U^{head} \cup U^{body} \cup U^{html}$

Abbildungen:

$\rightarrow_{URL} = \Im(url^{string:url}) =$

{<"http://www.cs.tu-berlin.de/~josefw/phd/index.html" , URL_1>,
<"http://www.cs.tu-berlin.de/~josefw/phd/introduction.html", URL_2>,
<"http://www.cs.tu-berlin.de/~josefw/phd/state.html", URL_3>,
<"http://www.cs.tu-berlin.de/~josefw/phd/query.html", URL_4>,
<"http://www.cs.tu-berlin.de/~josefw/phd/hynternetQL.html", URL_5>,
<"http://www.cs.tu-berlin.de/~josefw/phd/literature.html", URL_6>}

$\rightarrow_{TITLE} = \Im(title^{string:title}) =$

{<"Anfragesprachen für Internet-Informationssysteme", $TITLE_1$>,
<"Einleitung", $TITLE_2$>, <"Grundlagen", $TITLE_3$>,
<"Anfragen in Internet-Informationssystemen", $TITLE_4$>,
<"Anfragesprachen für Internet-Informationssysteme", $TITLE_5$>,

<"Literatur", $TITLE_6$>}

$\rightarrow_{H1} = \Im(h1^{string:h1}) =$

{<"Anfragesprachen für Internet-Informationssysteme“, $H1_1$>, <"Inhalt", $H1_2$>,
< "Einleitung", $H1_3$>, < "Grundlagen", $H1_4$>,
<"Anfragen in Internet-Informationssystemen", $H1_5$>,
<"Anfragesprachen für Internet-Informationssysteme", $H1_6$>, < "Literatur", $H1_7$>}

$\rightarrow_{LI} = \Im(li^{string:li}) =$

{<"Einleitung", LI_1>, <"Grundlagen", LI_2>,
<"Anfragen in Internet-Informationssystemen", LI_3, >,
<"Anfragesprachen für Internet-Informationssysteme", LI_4>, <"Literatur", LI_5>}

$\rightarrow_{UL} = \Im(ul^{li,li,li,li,li:ul}) = \{\langle LI_1, LI_2, LI_3, LI_4, LI_5, UL_1\rangle\}$

$\rightarrow_{HEAD} = \Im(head^{title:head}) =$

$\{\langle TITLE_1, HEAD_1\rangle, \langle TITLE_2, HEAD_2\rangle, \langle TITLE_3, HEAD_3\rangle, \langle TITLE_4, HEAD_4\rangle,$
$\langle TITLE_5, HEAD_5\rangle, \langle TITLE_6, HEAD_6\rangle\}$

$\rightarrow_{BODY1} = \Im(body1^{h1,h1,ul,pict,video:body}) = \{\langle H1_1, H1_2, UL_1, PICT_1, VIDEO_1, BODY_1\rangle\}$

$\rightarrow_{BODY2} = \Im(body2^{h1,string:body}) =$

{<$H1_3$, "Seit Entstehung des World Wide Web (WWW) hat sich ...", $BODY_2$>,
<$H1_4$, "Das Internet entstand Anfang der 70er Jahre als ...", $BODY_3$>,
<$H1_5$, "In den Internet-Informationssystemen stehen ...", $BODY_4$>,
<$H1_6$, "Wir behandeln die Entwicklung von Anfragesprachen ...", $BODY_5$>,
<$H1_7$, "Abiteboul, Beeri (1995) ...", $BODY_6$>}

$\rightarrow_{HTML} = \Im(html^{url,head,body:html}) =$

$\{\langle URL_1, HEAD_1, BODY_1, HTML_1\rangle, \langle URL_2, HEAD_2, BODY_2, HTML_2\rangle,$
$\langle URL_3, HEAD_3, BODY_3, HTML_3\rangle, \langle URL_4, HEAD_4, BODY_4, HTML_4\rangle,$
$\langle URL_5, HEAD_5, BODY_5, HTML_5\rangle, \langle URL_6, HEAD_6, BODY_6, HTML_6\rangle\}$

$\rightarrow = \rightarrow_{URL} \cup \rightarrow_{TITLE} \cup \rightarrow_{H1} \cup \rightarrow_{LI} \cup \rightarrow_{HEAD} \cup \rightarrow_{BODY1} \cup \rightarrow_{BODY2} \cup \rightarrow_{HTML}$

Relationen:

$UNIT = \Im(Unit^{html}) =$
$\{\langle HTML_1\rangle, \langle HTML_2\rangle, \langle HTML_3\rangle, \langle HTML_4\rangle, \langle HTML_5\rangle, \langle HTML_6\rangle\}$

$CONN^{li\times html} = \Im(Conn^{li,\ html}) =$
$\{\langle LI_1, HTML_2\rangle, \langle LI_2, HTML_3\rangle, \langle LI_3, HTML_4\rangle, \langle LI_4, HTML_5\rangle, \langle LI_5, HTML_6\rangle\}$

$SYMMETRIC = \Im(Symmetric) = \{\langle CONN^{li\times html}\rangle\} =$
$\{\langle\{\langle LI_1, HTML_2\rangle, \langle LI_2, HTML_3\rangle, \langle LI_3, HTML_4\rangle, \langle LI_4, HTML_5\rangle, \langle LI_5, HTML_6\rangle\}\rangle\}$

Frage 1: Welche HTML-Dokumente existieren ?

$\Im((\lambda\ x^{html})\ Unit^{html}(x^{html})) =$
$\{\langle U\rangle: \Im^U{}_x$ ist Modell von $Unit^{html}(x^{html})$ für $U \in \Im(html)\}=$
$\{\langle U\rangle: \langle\Im^U{}_x(x^{html})\rangle \in \Im^U{}_x(Unit^{html})$ für $U \in \mathcal{U}^{html}\}=$
$\{\langle U\rangle: \langle U\rangle \in$ UNIT für $U \in \mathcal{U}^{html}\}=$
$\{\langle HTML_1\rangle, \langle HTML_2\rangle, \langle HTML_3\rangle, \langle HTML_4\rangle, \langle HTML_5\rangle, \langle HTML_6\rangle\}$

Frage 2: Welche Teile der Stufe 2 hat das Objekt $html_1$?

$\Im((\lambda x)\ \nabla_2(html_1, x)) =$
$\{\langle U\rangle: \Im^U{}_x$ ist Modell von $\nabla_2(html_1, x)$ für $U \in \mathcal{U}\}=$
$\{\langle U\rangle: \Im^U{}_x$ ist Modell von $(\exists f_1, \exists x_1, ..., \exists x_n)\ \nabla_1(html_1, f_1(x_1, ..., x, ..., x_n))$ für $U \in \mathcal{U}\}=$
$\{\langle U\rangle: \Im^{U\ U1...Un\ FU1}{}_{x\ x1...xn\ f1}$ ist Modell von $\nabla_1(html_1, f_1(x_1, ..., x, ..., x_n))$

für mind. ein $FU_1 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n \in \mathit{U}$ und für $U \in \mathit{U}\}=$

$\{<U>: \Im^{U\ U1...Un\ FU1}_{x\ x1...xn\ f1}$ ist Modell von $(\exists f_2, \exists\ x_{21}, ..., \exists x_{2m})$

$Equal(html_1, f_2(x_{21}, ..., f_1(x_1, ..., x, ... , x_n), ..., x_{2m})$

für mind. ein $FU_1 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n \in \mathit{U}$ und für $U \in \mathit{U}\}=$

$\{<U>: \Im(html_1) =$

$\Im^{U\ U1...Un\ U21...Unm\ FU1\ FU2}_{x\ x1...xn\ x21...xnm\ f1\ f2}(f_2(x_{21}, ..., f_1(x_1, ..., x, ... , x_n), ..., x_{2m}))$

für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathit{U_{ij}}$ und für $U \in \mathit{U}\}=$

Bemerkung: wir lassen aus Übersichtlichkeitsgründen im Folgenden jeweils bei den Interpretationssymbolen die Interpretationsumgebung weg (hier: $\Im^{U\ U1...Un\ U21...Unm\ FU1\ FU2}_{x\ x1...xn\ x21...xnm\ f1\ f2}$)

$\{<U>: HTML_1 = \Im(f_2)(\Im(x_{21}), ..., \Im(f_1)(\Im(x_1), ..., \Im(x), ... , \Im(x_n)), ..., \Im(x_{2m})))$

für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathit{U}$ und für $U \in \mathit{U}\}=$

$\{<U>: HTML_1 = \rightarrow(U_{21}, ..., \rightarrow(U_1, ..., U, ... , U_n), ..., U_{2m}))$ für $U, U_i\ U_{jk} \in \mathit{U}\}=$

{<http://www.cs.tu-berlin.de/~josefw/phd/index.html">, $<H1_1>$, $<H1_2>$, $<UL_1>$, $<PICT_1>$, $<VIDEO_1>$}

Frage 3: Welches sind die Behälter der Stufe 2 von $title_1$? Oder anders ausgedrückt: Welche Objekte enthalten als Teil der Stufe 2 $title_1$?

$\Im((\lambda x)\ \triangle_2(title_1, x)) =$

$\{<U>: \Im^{U}_{x}$ ist Modell von $\triangle_2(title_1, x)$ für $U \in \mathit{U}\}=$

$\{<U>: \Im^{U}_{x}$ ist Modell von $(\exists f_1, \exists x_1, ..., \exists x_n)\ \triangle_1(f_1(x_1, ..., title_1, ... , x_n), x)$ für $U \in \mathit{U}\}=$

$\{<U>: \Im^{U\ U1\ ...\ Un\ FU1}_{x\ x1\ ...\ xn\ f1}$ ist Modell von $\triangle_1(f_1(x_1, ..., title_1, ... , x_n), x)$

für mind. ein $FU_1 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n \in \mathit{U}$ und für $U \in \mathit{U}\}=$

$\{<U>: \Im^{U\ U1\ ...\ Un\ FU1}_{x\ x1\ ...\ xn\ f1}$ ist Modell von $(\exists f_2, \exists\ x_{21}, ..., \exists x_{2m})$

$Equal(x, f_2(x_{21}, ..., f_1(x_1, ..., title_1, ... , x_n), ..., x_{2m})$

für mind. ein $FU_1 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n \in \mathit{U}$ und für $U \in \mathit{U}\}=$

$\{<U>: \Im^{U}_{x}(x) =$

$\Im^{U1\ ...\ Un\ U21...U2m\ FU1\ FU2}_{x1...xn\ x21...\ x2m\ f1\ f2}(f_2(x_{21}, ..., f_1(x_1, ..., title_1, ... , x_n), ..., x_{2m}))$

für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathit{U}$ und für $U \in \mathit{U}\}=$

Bemerkung: Es wird aus Übersichtlichkeitsgründen im Folgenden jeweils bei den Interpretationssymbolen die Interpretationsumgebung weggelassen (hier: $\Im^{U\ U1...Un\ U21...Unm\ FU1\ FU2}_{x\ x1...xn\ x21...xnm\ f1\ f2}$)

$\{<U>: \Im(x) = \Im(f_2)(\Im(x_{21}), ..., \Im(f_1)(\Im(x_1), ..., \Im(title_1), ... , \Im(x_n)), ..., \Im(x_{2m})))$

für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathit{U}$ und für $U \in \mathit{U}\}=$

$\{<U>: U = \rightarrow(U_{21}, ..., \rightarrow(U_1, ..., TITLE_1, ... , U_n), ..., U_{2m}))$ für $U, U_i\ U_{jk} \in \mathit{U}\}=$

$\{<HTML_1>\}$

Für die folgenden Anfragen **Frage 4**, **Frage 5** und **Frage 6**, die ähnlich wie die bisherigen Anfragen interpretiert werden, werden die die einzelnen Interpretationsschritte weggelassen.

Frage 4: Welche Objekte enthalten den Titel „Einleitung“ als Teil der Stufe 2 ?

$\Im((\lambda x)\ \triangle_2(title^{string:title}(\text{„Einleitung“}), x)) = \{<HTML_2>\}$

Frage 5: Welche HTML-Dokumente enthalten Header h1 ?
$\Im((\lambda x)\ \triangle_m(h1^{string:h1}(x^{string}), x)) =$
$\{<HTML_1>, <HTML_2>, <HTML_3>, <HTML_4>, <HTML_5>, <HTML_6>\}$

Frage 6: Welche HTML-Dokumente enthalten an erster Stelle im Body den Header h1("Einleitung") als Teil ?
$\Im((\lambda x)\ \exists x_1\ \triangle_m(body^{h1:body}(h1^{string:h1}(„Einleitung“), x_1), x)$
$\exists x_1\ \exists x_2\ \triangle_m(body^{h1:body}(h1^{string:h1}(„Einleitung“), x_1, x_2), x) \vee ... \vee$
$\exists x_1\ ...\ \exists x_n\ \triangle_m(body^{h1:body}(h1^{string:h1}(„Einleitung“), x_1, ..., x_n), x)) =$
$\{<HTML_2>\}$
Bemerkung: Die maximale Anzahl von Teilobjekten pro Stufe n wird bel. aber fest gesetzt.

Frage 7: Aus welchen Teilen bestehen die Dokumente, die das Teilobjekt $title_1$ enthalten ?
$\Im((\lambda x)\ \exists x_a\ \blacktriangle_m(title_1, x_a) \wedge \blacktriangledown_m(x_a, x)) =$
$\{<U>:$ $\Im^{U\ Ua}_{x\ xa}$ ist Modell von $\triangle_2(title_1, x_a)$ und $\Im^{U\ Ua}_{x\ xa}$ ist Modell von $\triangle_1(title_1, x_a)$ und
$\Im^{U\ Ua}_{x\ xa}$ ist Modell von $\nabla_2(x_a, x)$ und $\Im^{U\ Ua}_{x\ xa}$ ist Modell von $\nabla_1(x_a, x)$
für mind. ein $U_a \in \mathit{U}$ und für $U \in \mathit{U}\} =$
$\{<U>:$ $\Im^{U\ Ua}_{x\ xa}$ ist Modell von $(\exists f_1, \exists x_1, ..., \exists x_n)\ \triangle_1(f_1(x_1, ..., title_1, ... , x_n), x_a)$ und
$\Im^{Ua}_{xa}(x_a) = \Im^{U\ Ua\ U1...Un\ FU1}_{x\ xa\ x1...xn\ f1}(f_1(x_1, ..., title_1, ... , x_n)$ und
$\Im^{U\ Ua}_{x\ xa}$ ist Modell von $(\exists f_1, \exists x_1, ..., \exists x_n)\ \nabla_1(x_a, f_1(x_1, ..., x, ... , x_n))$ und
$\Im^{Ua}_{xa}(x_a) = \Im^{U\ U1...Un\ FU1}_{x\ x1...xnf1}(f_1(x_1, ..., x, ... , x_n)$
für mind. ein $FU_1 \in \{\rightarrow\}$ und für mind. ein $U_a, U_1, ..., U_n \in \mathit{U}$ und für $U \in \mathit{U}\} =$
$\{<U>:$ $\Im^{Ua\ U1\ ...\ Un\ FU1}_{xa\ x1\ ...\ xn\ f1}$ ist Modell von $\triangle_1(f_1(x_1, ..., title_1, ... , x_n), x_a)$ und
$\Im^{Ua}_{xa}(x_a) = \Im^{U\ Ua\ U1...Un\ FU1}_{x\ xa\ x1...xn\ f1}(f_1(x_1, ..., title_1, ... , x_n)$ und
$\Im^{Ua\ U1\ ...\ Un\ FU1}_{xa\ x1\ ...\ xn\ f1}$ ist Modell von $\nabla_1(x_a, f_1(x_1, ..., x, ... , x_n))$ und
$\Im^{Ua}_{xa}(x_a) = \Im^{U\ U1...Un\ FU1}_{x\ x1...xnf1}(f_1(x_1, ..., x, ... , x_n)$
für mind. ein $FU_1 \in \{\rightarrow\}$ und für mind. ein $U_a, U_1, ..., U_n \in \mathit{U}$ und für $U \in \mathit{U}\} =$
$\{<U>:$ $\Im^{Ua}_{xa}(x_a) =$
$\Im^{U1\ ...\ Un\ U21...U2m\ FU1\ FU2}_{x1...xn\ x21...\ x2m\ f1\ f2}(f_2(x_{21}, ..., f_1(x_1, ..., title_1, ... , x_n), ..., x_{2m}))$ und
$\Im^{Ua}_{xa}(x_a) = \Im^{U\ Ua\ U1...Un\ FU1}_{x\ xa\ x1...xn\ f1}(f_1(x_1, ..., title_1, ... , x_n)$ und
$\Im^{Ua}_{xa}(x_a) =$
$\Im^{U1\ ...\ Un\ U21...U2m\ FU1\ FU2}_{x1...xn\ x21...\ x2m\ f1\ f2}(f_2(x_{21}, ..., f_1(x_1, ..., x, ... , x_n), ..., x_{2m}))$ und
$\Im^{Ua}_{xa}(x_a) = \Im^{U\ U1...Un\ FU1}_{x\ x1...xn\ f1}(f_1(x_1, ..., x, ... , x_n)$
für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_a, U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathit{U}$
und für $U \in \mathit{U}\}=$
$\{<U>:$ $U_a = \rightarrow(U_{21}, ..., \rightarrow(U_1, ..., TITLE_1, ... , U_n), ..., U_{2m}))$ und
$U_a = \rightarrow(U_1, ..., TITLE_1, ... , U_n)$ und
$U_a = \rightarrow(U_{21}, ..., \rightarrow(U_1, ..., U, ... , U_n), ..., U_{2m}))$ und
$U_a = \rightarrow(U_1, ..., U, ... , U_n)$
für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_a, U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathit{U}$
und für $U \in \mathit{U}\}=$
$\{<"http://www.cs.tu-berlin.de/\sim josefw/phd/index.html">, <URL_1>, <HEAD_1>, <TITLE_1>, <BODY_1>, <H1_1>, <H1_2>, <UL_1>, <PICT_1>, <VIDEO_1>\}$
Bemerkung: Die maximale Anzahl von Stufen wird aus Übersichtlichkeitsgründen mit m=2 festgesetzt. Weiterhin werden aus demselben Grund einzelne Interpretationsschritte übersprungen.

Frage 8: Welche Geschwister hat das Teilobjekt li_1 ?

$\Im((\lambda\ x)\ \diamond(li_1, x)) =$

$\{<U>: \Im^{U}{}_{x}$ ist Modell von $\Diamond(li_1, x)$ für $U \in \mathcal{U}\} =$

$\{<U>: \Im^{U}{}_{x}$ ist Modell von $\exists f, \exists x_1\ Equal(f(li_1, x, ..., x), x_1) \wedge Equal(f(x, li_1, ..., x), x_1) \wedge ... \wedge$
$Equal(f(x, x, ..., li_1), x_1)$ für $U \in \mathcal{U}\} =$

$\{<U>: \Im^{U\ U1\ FU}{}_{x\ x1\ f}$ ist Modell von $Equal(f(li_1, x, ..., x), x_1) \wedge Equal(f(x, li_1, ..., x), x_1) \wedge ... \wedge$
$Equal(f(x, x, ..., li_1), x_1)$ für mind. ein $FU \in \{\rightarrow\}$, für mind. ein $U_1 \in \mathcal{U}$, für $U \in \mathcal{U}\} =$

$\{<U>: \Im^{U\ U1\ FU}{}_{x\ x1\ f}$ ist Modell von $Equal(f(li_1, x, ..., x), x_1)$ und $\Im^{U\ U1\ FU}{}_{x\ x1\ f}$ ist Modell von
$Equal(f(x, li_1, ..., x), x_1)$ und ... und $\Im^{U\ U1\ FU}{}_{x\ x1\ f}$ ist Modell von $Equal(f(x, x, ..., li_1), x_1)$

für mind. ein $FU \in \{\rightarrow\}$, für mind. ein $U_1 \in \mathcal{U}$, für $U \in \mathcal{U}\} =$

$\{<U>: \Im^{U\ U1\ FU}{}_{x\ x1\ f}(f(li_1, x, ..., x)) = \Im^{U\ U1\ FU}{}_{x\ x1\ f}(x_1)$ und
$\Im^{U\ U1\ FU}{}_{x\ x1\ f}(f(x, li_1, ..., x)) = \Im^{U\ U1\ FU}{}_{x\ x1\ f}(x_1)$ und ... und
$\Im^{U\ U1\ FU}{}_{x\ x1\ f}(f(x, x, ..., li_1)) = \Im^{U\ U1\ FU}{}_{x\ x1\ f}(x_1)$ für mind. ein $FU \in \{\rightarrow\}$und
für mind. ein $U_1 \in \mathcal{U}$, für $U \in \mathcal{U}\} =$

$\{<U>: \rightarrow(LI_1, U, ..., U)) = U_1$ und $\rightarrow(U, LI_1, ..., U)) = U_1$ und ... und $\rightarrow(U, U, ..., LI_1)) = U_1$
für mind. ein $FU \in \{\rightarrow\}$, für mind. ein $U_1 \in \mathcal{U}$, für $U \in \mathcal{U}\} =$

$\{<LI_2>, <LI_3>, <LI_4>, <LI_5>\}$

Frage 9: Welche Beziehungsart existiert zwischen li und html ?

$\Im((\lambda\ P^{li,\ html})\ (\exists x^{li}{}_1\ \exists x^{html}{}_1\ P^{li,\ html}{}_1(x^{li}{}_1, x^{html}{}_2)) =$

$\{<rel>: \Im^{rel}{}_{P}$ ist Modell von $(\exists x^{li}{}_1\ \exists x^{html}{}_2\ P^{li,\ html}{}_1(x^{li}{}_1, x^{html}{}_2))\ \}=$

$\{<rel>: \Im^{rel\ Uli\ Uhtml}{}_{P\ x1\ x2}$ ist Modell von $P^{li,html}{}_1(x^{li}{}_1, x^{html}{}_1)$
für mindestens ein $LI_1 \in \mathcal{U}^{li}$, $HTML_1 \in \mathcal{U}^{html}\}=$

$\{<rel>:< \Im^{rel\ Uli\ Uhtml}{}_{P\ x1\ x2}(x^{li}{}_1), \Im^{rel\ LI\ HTML}{}_{P\ x1\ x2}(x^{html}{}_1)> \in \Im^{rel\ LI\ HTML}{}_{P\ x1\ x2}(P^{li,\ html})$
für mindestens ein $LI_1 \in \mathcal{U}^{li}$, $HTML_1 \in \mathcal{U}^{html}\}=$

$\{<rel>: <LI, HTML> \in CONN^{li \times html}$ für mindestens ein $LI_1 \in \mathcal{U}^{li}$, $HTML_1 \in \mathcal{U}^{html}\}=$

$\{<\{<LI_1, UNIT_2>, <LI_2, UNIT_3>, <LI_3, UNIT_4>, <LI_4, UNIT_5>, <LI_5, UNIT_6>\} >\}$

Frage 10: Welche Beziehungsarten haben die Eigenschaft Symmetric ?

$\Im((\lambda\ P^{s1,\ s2})\ Symmetric(P^{s1,\ s2})) =$

$\{<rel> : \Im^{rel}{}_{P}$ ist Modell von $Symmetric(P^{s1,\ s2})\} =$

$\{<rel> : <\Im^{rel}{}_{P}(P^{s1,\ s2})> \in \Im^{rel}{}_{P}(Symmetric)\} =$

$\{\ <CONN^{li \times html}>\}$

4.5 StructuredQL: Eine Anfragesprache für strukturierte Dokumente

Anwender von StructuredQL sind Benutzer, die eine universelle Suche in strukturierten Dokumenten durchführen möchten. Mit StructuredQL werden SimpleStructuredQL, SimpleLinkQL, LinkQL und Nested&LinkQL zusammengeführt.

StructuredQL ist eine prädikatenlogische Sprache 2. Stufe. Sie enthält Quantoren (Existenz- und Allquantor) für Funktionen und Prädikate, den Lambda-Operator für Prädikate und Prädikatenkonstanten der zweiten Stufe.

4.5.1 Syntax

Sorten

1. mail, person host, file, news, menu, unit, string, date, int, regExpr sind Sorten.
2. bool, html, head, body, title, chapter, abstract, ... sind Sorten.
3. Das sind alle Sorten.
Im Folgenden werden die Variablen s, s_1, ..., s_n für diese Sorten benutzt.

Alphabet
1. Individuenkonstanten für Dokumente: mail, $mail_1$, ... , person, $person_1$, ... , host, $host_1$, ... , file, $file_1$, ... , news, $news_1$, ... , menu, $menu_1$, ... , unit, $unit_1$, ... , string, $string_1$, ..., date, $date_1$, ..., 1, 2, ... true, false, html, $html_1$, ..., head, $head_1$, ... body, $body_1$, ..., title, $title_1$, ..., chapter, $chapter_1$, ... abstract, $abstract_1$, ..., ...
2. Individuenvariablen für Dokumente: x^{mail}, $x^{mail}{}_1$, ... , x^{person}, $x^{person}{}_1$, ... , x^{host}, $x^{host}{}_1$, ... , x^{file}, $x^{file}{}_1$, ... , x^{news}, $x^{news}{}_1$, ... , x^{menu}, $x^{menu}{}_1$, ... , x^{unit}, $x^{unit}{}_1$, ... , x^{string}, $x^{string}{}_1$, ... , x^{date}, $x^{date}{}_1$, ... , x^{int}, $x^{int}{}_1$, ..., $x^{regExpr}$, $x^{regExpr}{}_1$, ... , x^{bool}, $x^{bool}{}_1$, ... , x^{html}, $x^{html}{}_1$, ... , x^{head}, $x^{head}{}_1$, ... , x^{body}, $x^{body}{}_1$, ... , x^{title}, $x^{title}{}_1$, ..., $x^{chapter}$, $x^{chapter}{}_1$, ..., $x^{abstract}$, $x^{abstract}{}_1$, ..., x, x_1, ... , ...
3. Individuenkonstanten für eine Stufe: 1, 2, 3, ...
4. Individuenvariable für eine Stufe: m, s
5. Funktionskonstanten von Sorten s_1, ..., s_n auf die angegebene Sorte: $html^{s:html}$, $head^{s1,..., sn:head}$, $body^{s1, ..., sn:body}$, $title^{s1, ...,sn:title}$, $chapter^{s1, ...,sn:chapter}$, $abstract^{s1, ...,sn:abstract}$, ...
6. Funktionsvariablen von Sorten s_1, ..., s_n auf die Sorte s: $f^{s1, ..., sn:s}{}_1$, $f^{s1, ...,sn:s}{}_2$, ...
7. Prädikatenkonstanten verschiedener Sorten (wenn nicht anders angegeben 2-stellig und wenn nicht explizit angegeben haben die Prädikate die Sorten des jeweils ersten Prädikats):
a) Artikel: $Content^{news,string}$, $Date^{news,date}$, Group-id, $Length^{file,int}$, Newsgroup, Subject, Title, User-id, ...
b) Dateien: $Contains^{file,file}$, $Content^{file,string}$, $Date^{file,date}$, $File\text{-}name^{file,string}$, $File\text{-}type^{file,string}$, $Group\text{-}id^{file,string}$, $Length^{file,int}$, $Title^{file,string}$, $User\text{-}id^{file,string}$, ...
c) Dokumenthinweise: $Abstract^{unit,string}$, Creator, Contributors, Coverage, $Conn^{unit,unit}$, $Date^{unit,date}$, Database, Description, Format, Identifier, Language, Publisher, Relation, Rights, Source, Subject, Type, Title, ...
d) E-Mails: $Action^{mail,string}$, Copy, Content-type, Content-transfer-encoding, Content-id, Content-description, $Date^{mail,date}$, Expires, From, In-reply-to, Message-id, Mime-version, Prece-de(n)ce, Priority, Received, Reply-to, Subject, To, ...
e) Menüs: $Content^{menu,string}$, $Date^{menu,date}$, Group-id, $Length^{file,int}$, Subject, Title, User-id, ...
f) Personen/Gruppen: $Communication\text{-}address\text{-}private^{person,string}$, Communication-address-office, $Date^{person,date}$, Domain-name, Fields-of-business, Information-address, Keywords, Members, Name-private, Name-headquarters, E-Mail-private, E-mail-office, Fax-private, Fax-office, Phone-private, Phone-office, Postal-address-private, Postal-address-office, Personal-data, Public-key, Projects, Title, Type, ...
g) Rechner: Address-administrator ^{host,string}, Address-technical-person, $Date\text{-}of\text{-}last\text{-}modification^{host,date}$, E-mail-administrator, , E-mail-technical-person, Host-domain-name, Host-type, Information-address, Ip, Name-administrator, Name-technical-person, Nameservers-of-network, Operating-system, Phone-administrator, Phone-technical-person, ...
h) Vergleichsprädikate:
h1) Zahlenvergleiche: $\neq^{date,date}$, $<^{date,date}$, $>^{date,date}$, $\leq^{date,date}$, $\geq^{date,date}$, $\neq^{int,int}$, $<^{int,int}$, $>^{int,int}$, $\leq^{int,int}$, $\geq^{int,int}$
h2) Vergleich von Zeichenfolgen: $Contains^{regExpr,string}$, $Near^{string,int,string}$
h3) Linguistische Operationen: $Upper^{string,string}$, $Lower^{string,string}$, $Stem^{string,string}$, $Fuzzy^{string,string}$, $Soundex^{string,string}$
h4) Thesaurusoperatoren: $Syn^{string,string}$, $NT^{string,string}$, $BT^{string,string}$, $PT^{string,string}$
i) Netzwerkfunktionen: Path (n-stellig), $\downarrow$, $\uparrow$, $\Downarrow$, $\Uparrow$, $\Leftrightarrow$, Θ (dreistellig)

j) Dokumente (einstellig): $Unit^{s}$
k) Beziehungen (zweistellig): $Conn^{s1, s2}$
l) Gleichheit (zweistellig): $Equal^{s1, s2}$
m) Teile/Behälter: $\triangledown$, $\blacktriangledown$, $\triangle$, $\blacktriangle$, $\diamond$
8. Prädikatenvariablen (2-stellig) verschiedener Sorten: $P^{s1, s2}{}_{1}$, $P^{s1, s2}{}_{2}$, ...
9. Prädikatenkonstanten 2. Stufe: Reflexive, Nonreflexive, Irreflexive, Symmetric, Asymmetric, Antisymmetric, Transitive, NegativelyTransitive, Nontransitive, EquivalenceRelation, Complete, StronglyComplete, ...
10. Logische Symbole: $\neg$ (nicht), $\wedge$ (und), $\vee$ (oder), $\rightarrow$ (Implikation) $\leftrightarrow$ (Äquivalenz)
$\exists$ (es gibt ein), $\forall$ (für alle), λ (alle, die)
11. Technische Symbole: (,)
Mit diesen Symbolen werden induktiv die Terme und Formeln gebildet:

Terme:
1. Individuenkonstanten und -variablen der Sorte s sind Terme der Sorte s.
2. Wenn t_1 Term der Sorte s_1 , t_2 Term der Sorte s_2 ,... , t_n Term der Sorte s_n ist, und f ein Funktionssymbol (Konstante oder Variable) von Sorten s_1 ,... , s_n auf s ist, dann ist
$f^{s1, \ldots, sn:s}(t_1, t_2, \ldots , t_n)$ Term der Sorte s.
3. Das sind alle Terme.

Formeln
1. Wenn x_1 Individuenkonstante oder -variable der Sorte s_1 ist und x_2 Individuenkonstante oder -variable der Sorte s_2 ist und P eine Prädikatenkonstante oder -variable der Sorten s_1, s_2 ist, dann ist $P^{s1,s2}(x_1, x_2)$ eine Formel.
2. Wenn $x^{s1}{}_{1}, \ldots, x^{s1}{}_{n}, x^{s\alpha}{}_{\alpha}, x^{s\beta}{}_{\beta}, x^{s\gamma}{}_{\gamma}$ Individuenkonstanten oder -variablen für Dokumente sind , P eine Prädikatenkonstante oder -variable der Sorten s_1, s_2 ist und s eine Individuenkonstante oder -variable für eine Stufe ist, dann sind
Pfad zwischen $x^{s1}{}_{1}$ und $x^{s1}{}_{n}$ bzgl. P: $Path^{P}(x^{s1}{}_{1}, \ldots, x^{s1}{}_{n})$,
Nachfolger von $x^{s\alpha}{}_{\alpha}$ der Stufe s bzgl. P: $\downarrow^{P}{}_{s}(x^{s\alpha}{}_{\alpha}, x^{s\beta}{}_{\beta})$,
Vorgänger von $x^{s\alpha}{}_{\alpha}$ der Stufe s bzgl. P: $\uparrow^{P}{}_{s}(x^{s\alpha}{}_{\alpha}, x^{s\beta}{}_{\beta})$,
Nachfolger von $x^{s\alpha}{}_{\alpha}$ bis zur Stufe s bzgl. P: $\Downarrow^{P}{}_{s}(x^{s\alpha}{}_{\alpha}, x^{s\beta}{}_{\beta})$,
Vorgänger von $x^{s\alpha}{}_{\alpha}$ bis zur Stufe s bzgl. P: $\Uparrow^{P}{}_{s}(x^{s\alpha}{}_{\alpha}, x^{s\beta}{}_{\beta})$,
Geschwister von $x^{s\alpha}{}_{\alpha}$ bzgl. P: $\Leftrightarrow^{P}(x^{s\alpha}{}_{\alpha}, x^{s\beta}{}_{\beta})$,
Stationen zwischen $x^{s\alpha}{}_{\alpha}$ und $x^{s\beta}{}_{\beta}$ bzgl. P: $\Theta^{P}(x^{s\alpha}{}_{\alpha}, x^{s\beta}{}_{\beta}, x^{s\gamma}{}_{\gamma})$ Formeln.
4. Wenn t Term der Sorte s, t_1 Term der Sorte s_1 , t_2 Term der Sorte s_2 ,... , t_n Term der Sorte s_n ist, m eine Stufe (Konstante oder Variable) ist, dann sind
$Unit^{s}(t)$, $Conn^{s1, s2}(t_1, t_2)$, $P^{s1, s2}(t_1, t_2)$, $Equal^{s1, s2}(t_1, t_2)$,
Teile von t_1 der Stufe m: $\triangledown_{m}(t, x)$
Teile von t_1 bis zur Stufe m: $\blacktriangledown_{m}(t, x)$
Behälter von t_1 der Stufe m: $\triangle_{m}(t, x)$
Behälter von t_1 bis zur Stufe m: $\blacktriangle_{m}(t, x)$
Geschwister von t_1: $\diamond(t_1, x)$ sind Formeln.
5. Wenn F, F_1 und F_2 Formeln sind und x^{s} eine Individuenvariable der Sorte s ist, dann sind auch $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \rightarrow F_2$, $F_1 \leftrightarrow F_2$, $\exists x^{s} F$, $\forall x^{s} F$, $\exists P^{s1,s2} F$ und $\forall P^{s1,s2} F$ Formeln.
6. Wenn $P^{s1, s2}{}_{1}$ ein Prädikatensymbol (Konstante oder Variable) der Sorten s_1, s_2 ist und Prop eine Prädikatenkonstante zweiter Stufe für Eigenschaften von Relationen ist, dann ist auch $Prop(P^{s1, s2}{}_{1})$ eine Formel.
7. Das sind alle Formeln.

Anfragen

1. Wenn F eine Formel und $x^{s1}{}_{1} \ldots x^{sn}{}_{n}$ Individuenvariablen der Sorten $s_1, \ldots, s_n$ sind, dann ist $(\lambda\, x^{s1}{}_{1} \ldots x^{sn}{}_{n})$ F eine Anfrage.
2. Wenn F eine Formel und $P^{s1,\ldots,sn}$ eine Prädikatenvariable der Sorten $s_1, \ldots, s_n$ ist, dann ist $(\lambda\, P^{s1,\ldots,sn})$ F eine Anfrage.
3. Das sind alle Anfragen.

4.5.2 Semantik

Es werden nichtleere Individuenbereiche unterschiedlicher Sorte zugrundegelegt:
U^{mail}, U^{person}, U^{host}, U^{file}, U^{news}, U^{menu}, U^{unit}, U^{html}, U^{head}, U^{body}, U^{title}, $U^{chapter}$, $U^{abstract}$, $U^{heading}$, U^{ol}, U^{ul}, U^{li}, U^{string}, $U^{regExpr}$, U^{int}, U^{bool}, U^{date}, ...
Für Variablen bel. Sorte wird die Vereinigungsmenge aller nichtleeren Individuenbereiche U zugrundegelegt. Für Funktionen werden Abbildungen und für Prädikate Relationen zugrundegelegt.

Interpretation der Individuensymbole

$\Im(news_i) \in U^{news}$, $\Im(file_i) \in U^{file}$, $\Im(unit_i) \in U^{unit}$, $\Im(mail_i) \in U^{mail}$, $\Im(menu_i) \in U^{menu}$, $\Im(person_i) \in U^{person}$, $\Im(host_i) \in U^{host}$, $\Im(string_i) \in U^{string}$, $\Im(date_i) \in U^{date}$, $\Im(regExpr_i) \in U^{regExpr}$, $\Im(1) \in U^{int}$, $\Im(2) \in U^{int}$, $\Im(true) \in U^{bool}$, $\Im(false) \in U^{bool}$, $\Im(html_i) \in U^{html}$, $\Im(head_i) \in U^{head}$, $\Im(body_i) \in U^{body}$, $\Im(title_i) \in U^{title}$, $\Im(chapter_i) \in U^{chapter}$, $\Im(abstract_i) \in U^{abstract}$, ... ,
für i = 1, ..., n (i kann auch weggelassen werden)
$\Im(x^{news}{}_{i}) \in U^{news}$, $\Im(x^{file}{}_{i}) \in U^{file}$, $\Im(x^{unit}{}_{i}) \in U^{unit}$, $\Im(x^{mail}{}_{i}) \in U^{mail}$, $\Im(x^{menu}{}_{i}) \in U^{menu}$, $\Im(x^{person}{}_{i}) \in U^{person}$, $\Im(x^{host}{}_{i}) \in U^{host}$, $\Im(x^{string}{}_{i}) \in U^{string}$, $\Im(x^{date}{}_{i}) \in U^{date}$, $\Im(x^{regExpr}{}_{i}) \in U^{regExpr}$, $\Im(x^{int}{}_{i}) \in U^{int}$, $\Im(x^{bool}{}_{i}) \in U^{bool}$, $\Im(x^{html}{}_{i}) \in U^{html}$, $\Im(x^{head}{}_{i}) \in U^{head}$, $\Im(x^{body}{}_{i}) \in U^{body}$, $\Im(x^{title}{}_{i}) \in U^{title}$, $\Im(x^{chapter}{}_{i}) \in U^{chapter}$, $\Im(x^{abstract}{}_{i}) \in U^{abstract}$, ..., $\Im(x_i) \in U$ für i = 1, ..., n (i kann auch weggelassen werden)

Interpretation der Funktionssymbole

$\Im(html^{s:html}) \in \{U^{s} \rightarrow_{HTML} U^{html}\}$, $\Im(head^{s1,\ \ldots,\ sn:head}) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow_{HEAD} U^{head}\}$, $\Im(body^{s1,\ \ldots,\ sn:body}) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow_{BODY} U^{body}\}$, $\Im(title^{s1,\ \ldots,sn:title}) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow_{TITLE} U^{title}\}$, $\Im(chapter^{s1,\ \ldots,sn:chapter}) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow_{CHAPTER} U^{chapter}\}$, $\Im(abstract^{s1,\ \ldots,sn:abstract}) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow_{ABSTRACT} U^{abstract}\}$, ... ,
$\Im(f^{s1,\ \ldots\ ,\ sn:sm}{}_{i}) \in \{U^{s1} \times \ldots \times U^{sn} \rightarrow \Im(s_m)\}$,

Interpretation der Prädikatensymbole

$\Im(UNIT^{s}) \subseteq U^{s}$
K sei eine zweistellige Prädikatenkonstante.
$\Im(K^{s1,s2}) \subseteq U^{s1} \times U^{s2}$
$\Im(P^{s1,s2}) \subseteq U^{s1} \times U^{s2}$

Interpretation der Terme

k sei Funktionskonstante
$\Im(k^{s1,\ \ldots\ ,\ sn:s}{}_{i}(t_1, \ldots , t_n)) = \Im(k^{s1,\ \ldots\ ,\ sn:s}{}_{i})\ (\Im(t_1), \Im(t_2), \ldots , \Im(t_n))$
$\Im(f^{s1,\ \ldots\ ,\ sn:s}{}_{i}(t_1, \ldots , t_n)) = \Im(f^{s1,\ \ldots\ ,\ sn:s}{}_{i})\ (\Im(t_1), \Im(t_2), \ldots , \Im(t_n))$

Interpretation der Formeln

1. $\Im$ ist Modell von $Unit^s(t)$ gdw $<\Im(t)> \in \Im(Unit^s)$
$\Im$ ist Modell von $Conn^{s1, s2}(t_1, t_2)$ gdw $<\Im(t_1), \Im(t_2)> \in \Im(Conn^{s1, s2})$
$\Im$ ist Modell von $Equal(t_1, t_2)$ gdw $\Im(t_1) = \Im(t_2)$
$\Im$ ist Modell von $P^{s1, s2}(t_1, t_2)$ gdw $<\Im(t_1), \Im(t_2)> \in \Im(P^{s1, s2})$
$\Im$ ist Modell von $\neq^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist ungleich $\Im(x^{s2}_j)$
$\Im$ ist Modell von $<^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist kleiner als $\Im(x^{s2}_j)$
$\Im$ ist Modell von $>^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist größer als $\Im(x^{s2}_j)$
$\Im$ ist Modell von $\leq^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist ist kleiner gleich $\Im(x^{s2}_j)$
$\Im$ ist Modell von $\geq^{s1,s2}(x^{s1}_i, x^{s2}_j)$ gdw $\Im(x^{s1}_i)$ ist ist größer gleich $\Im(x^{s2}_j)$
$\Im$ ist Modell von $Contains^{regExpr,string}(x^{regExpr}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ enthält den regulären Ausdruck $\Im(x^{regExpr}_1)$
$\Im$ ist Modell von $Near^{string,int,string}(x^{string}_1, x^{int}_2, x^{string}_1)$ gdw
    $\Im(x^{string}_3)$ befindet sich in der $\Im(x^{int}_2)$ - Zeichenzahl-Umgebung von $\Im(x^{string}_1)$
$\Im$ ist Modell von $Upper^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist die in Großbuchstaben gewandelte Zeichenfolge von $\Im(x^{string}_1)$
$\Im$ ist Modell von $Lower^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist die in Kleinbuchstaben gewandelte Zeichenfolge von $\Im(x^{string}_1)$
$\Im$ ist Modell von $Stem^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist die Stammform von $\Im(x^{string}_1)$
$\Im$ ist Modell von $Fuzzy^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist rechschreibähnlich zu $\Im(x^{string}_1)$
$\Im$ ist Modell von $Soundex^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist phonetisch ähnlich zu $\Im(x^{string}_1)$
$\Im$ ist Modell von $Syn^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist synonym zu $\Im(x^{string}_1)$
$\Im$ ist Modell von $NT^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist Unterbegriff von $\Im(x^{string}_1)$
$\Im$ ist Modell von $BT^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist Oberbegriff von $\Im(x^{string}_1)$
$\Im$ ist Modell von $PT^{string,string}(x^{string}_1, x^{string}_2)$ gdw
    $\Im(x^{string}_2)$ ist Vorzugsbegriff von $\Im(x^{string}_1)$
2. $\Im$ ist Modell von $Path^{Conn}(x_1, ..., x_n)$ gdw
    $\Im$ ist Modell von $Conn(x_1, x_2)$ und $\Im$ ist Modell von $Conn(x_2, x_3)$ und ... und
    $\Im$ ist Modell von $Conn(x_{n-1}, x_n)$ und $\Im(x_1) \neq \Im(x_2) \neq ... \neq \Im(x_n)$
$\Im$ ist Modell von $\downarrow^{Conn}_s(x_\alpha, x_\beta)$ gdw $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_1, x_2, x_3, ..., x_{s-1}, x_\beta)$
$\Im$ ist Modell von $\uparrow^{Conn}_s(x_\alpha, x_\beta)$ gdw $\Im$ ist Modell von $\downarrow^{Conn}_s(x_\beta, x_\alpha)$
$\Im$ ist Modell von $\Downarrow^{Conn}_s(x_\alpha, x_\beta)$ gdw
    $\Im$ ist Modell von $\downarrow^{Conn}_s(x_\alpha, x_\beta)$ oder
    $\Im$ ist Modell von $\downarrow^{Conn}_{s-1}(x_\alpha, x_\beta)$ oder ... oder
    $\Im$ ist Modell von $\downarrow^{Conn}_1(x_\alpha, x_\beta)$
$\Im$ ist Modell von $\Uparrow^{Conn}_s(x_\alpha, x_\beta)$ gdw $\Im$ ist Modell von $\Downarrow^{Conn}_s(x_\beta, x_\alpha)$
$\Im$ ist Modell von $\Leftrightarrow^{Conn}(x_\alpha, x_\beta)$ gdw
    $\Im$ ist Modell von $Conn(x_\gamma, x_\alpha)$ und $\Im$ ist Modell von $Conn(x_\gamma, x_\beta)$ und $\Im(x_\alpha) \neq \Im(x_\beta)$
$\Im$ ist Modell von $\Theta^{Conn}(x_\alpha, x_\beta, x_\gamma)$ gdw
    $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_\gamma, x_2, x_3, ..., x_n, x_\beta)$ und
    $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_1, x_\gamma, x_3, ..., x_n, x_\beta)$ und ... und
    $\Im$ ist Modell von $Path^{Conn}(x_\alpha, x_1, x_2, x_3, ..., x_\gamma, x_\beta)$ wobei n bel. aber fest

3. f sei Funktion (Konstante oder Variable) bel. Sorte und bel. aber fester Stellenzahl

$\Im$ ist Modell von $\nabla_m(t, x)$ gdw

$\Im$ ist Modell von $(\exists f, \exists x_1, ..., \exists x_n)\ \nabla_{m-1}(t, f(x_1, ..., x, ..., x_n))$

$\Im$ ist Modell von $\nabla_1(t, x)$ gdw

$\Im$ ist Modell von $(\exists f, \exists x_1, ..., \exists x_n)$ Equal$(t, f(x_1, ..., x, ..., x_n))$

$\Im$ ist Modell von $\blacktriangledown_m(t, x)$ gdw

$\Im$ ist Modell von $\nabla_m(t, x)$ und $\Im$ ist Modell von $\nabla_{m-1}(t, x)$ und ... und
$\Im$ ist Modell von $\nabla_1(t, x)$

$\Im$ ist Modell von $\triangle_m(t, x)$ gdw

$\Im$ ist Modell von $(\exists f, \exists x_1, ..., \exists x_n)\ \triangle_{m-1}(f(x_1, ..., t, ..., x_n), x)$

$\Im$ ist Modell von $\triangle_1(t, x)$ gdw

$\Im$ ist Modell von $(\exists f, \exists x_1, ..., \exists x_n)$ Equal$(x, f(x_1, ..., t, ..., x_n))$

$\Im$ ist Modell von $\blacktriangle_m(t, x)$ gdw

$\Im$ ist Modell von $\triangle_m(t, x)$ und $\Im$ ist Modell von $\triangle_{m-1}(t, x)$ und ... und
$\Im$ ist Modell von $\triangle_1(t, x)$

$\Im$ ist Modell von $\diamond(t, x)$ gdw

$\Im$ ist Modell von $\exists f, \exists x_1$ Equal$(f(t, x, ..., x), x_1) \wedge$ Equal$(f(x, t, ..., x), x_1) \wedge ... \wedge$
Equal$(f(x, x, ..., t), x_1)$

4. $\Im$ ist Modell von $\neg F$ gdw $\Im$ ist nicht Modell von F.

$\Im$ ist Modell von $(F_1 \wedge F_2)$ gdw $\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.

$\Im$ ist Modell von $(F_1 \vee F_2)$ gdw $\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.

$\Im$ ist Modell von $(F_1 \rightarrow F_2)$ gdw $\Im$ ist Modell von F2 wenn $\Im$ ist Modell von F_1.

$\Im$ ist Modell von $(F_1 \leftrightarrow F_2)$ gdw ($\Im$ ist Modell von F1 gdw $\Im$ ist Modell von F_2).

$\Im$ ist Modell von $(\exists\ x^s_i)$ F gdw $\Im^U_{xi}$ ist Modell von F für mindestens ein U $\in \mathcal{U}^s$.

$\Im$ ist Modell von $(\forall\ x^s_i)$ F gdw $\Im^U_{xi}$ ist Modell von F für alle U $\in \mathcal{U}^s$.

$\Im$ ist Modell von $(\exists\ f^{s1, ..., sn:s})$ F gdw

$\Im^{FU}_f$ ist Modell von F für mindestens ein FU $\in \{\mathcal{U}^{s1} \times ... \times \mathcal{U}^{sn} \rightarrow \mathcal{U}^s\}$.

$\Im$ ist Modell von $(\forall\ f^{s1, ..., sn:s})$ F gdw

$\Im^{FU}_f$ ist Modell von F für alle FU $\in \{\mathcal{U}^{s1} \times ... \times \mathcal{U}^{sn} \rightarrow \mathcal{U}^s\}$.

$\Im$ ist Modell von $(\exists$ f) F gdw $\Im^{FU}_f$ ist Modell von F für mindestens ein FU $\in \{ \rightarrow \}$.

$\Im$ ist Modell von $(\forall$ f) F gdw $\Im^{FU}_f$ ist Modell von F für alle FU $\in \{ \rightarrow \}$.

$\Im$ ist Modell von $(\exists\ P^{s1, s2})$ F gdw

$\Im^{rel}_P$ ist Modell von F für mindestens eine Relation über $\mathcal{U}^{s1} \times \mathcal{U}^{s2}$.

$\Im$ ist Modell von $(\forall\ P^{s1, s2})$ F gdw

$\Im^{rel}_P$ ist Modell von F für alle Relationen über $\mathcal{U}^{s1} \times \mathcal{U}^{s2}$.

$\Im$ ist Modell von Prop$(P^{s1, s2})$ gdw $<\Im(P^{s1, s2})> \in \Im$(Prop)

Prop sei Prädikatenkonstante 2. Stufe

Interpretation der Anfragen

5. $\Im((\lambda\ x^{s1}_1 ... x^{sn}_n)\ F) =$

$\{<U_1, ..., U_n>: \Im^{U1...Un}_{x1...xn}$ ist Modell von F für $U_1 \in \mathcal{U}^{s1}, ..., U_n \in \mathcal{U}^{sn}\}$.

$\Im((\lambda\ P^{s1, s2})\ F) = \{<rel>: \Im^{rel}_P$ ist Modell von F$\}$.

4.5.3 Beispiele

Es werden z.T. Individuenbereiche, Abbildungen und Relationen aus den vorangegangenen Beispielen verwendet:

Individuenbereiche:
U^{file}, U^{news} (aus Kapitel 4.1.3),
U^{string}, $U^{picture}$, U^{video}, U^{url}, U^{title}, U^{li}, U^{ul}, U^{head}, U^{body}, U (aus Kapitel 4.4.3),
U^{h1} = {$H1_1$, $H1_2$, $H1_3$, $H1_4$, $H1_5$, $H1_6$, $H1_7$, $H1_8$},
$U^{abstract}$ = {$ABSTRACT_1$},
U^{html} = {$HTML_1$, $HTML_2$, $HTML_3$, $HTML_4$, $HTML_5$, $HTML_6$, $HTML_7$, $HTML_8$, $HTML_9$, $HTML_{10}$, $HTML_{11}$, $HTML_{12}$, $HTML_{13}$, $HTML_{14}$, $HTML_{15}$, $HTML_{16}$, $HTML_{17}$, $HTML_{18}$}
$U^{unit} = U^{html} \cup U^{abstract}$

Abbildungen :
$\rightarrow$, $\rightarrow_{URL}$, $\rightarrow_{TITLE}$, $\rightarrow_{LI}$, $\rightarrow_{HEAD}$, $\rightarrow_{BODY1}$, $\rightarrow_{BODY2}$, $\rightarrow_{HTML}$(aus Kapitel 4.4.3),
$\rightarrow_{H1} = \Im(h1^{string:h1})$ =
{<"Anfragesprachen für Internet-Informationssysteme“, $H1_1$>, <"Inhalt", $H1_2$>,
< "Einleitung", $H1_3$>, < "Grundlagen", $H1_4$>,
<"Anfragen in Internet-Informationssystemen", $H1_5$>,
<"Anfragesprachen für Internet-Informationssysteme", $H1_6$>, <"Literatur", $H1_7$>,
<"Abstract", $H1_8$>}
$\rightarrow_{ABSTRACT} = \Im(abstract^{h1,string:abstract})$ =
{<$H1_8$, "Es wird zunächst ein Überblick ...", $ABSTRACT_1$>,

Relationen:
SUBJECT, $DATE^{unit \times date}$, LANGUAGE, FILENAME, CONTAINS, NEWSGROUP, $CONTENT^{news \times string}$ (aus Kapitel 4.1.3),
CONN (aus Kapitel 4.2.3),
USED-MATERIAL (aus Kapitel 4.3.3),
UNIT, $CONN^{li \times html}$, SYMMETRIC (aus Kapitel 4.4.3)
Zusätzlich enthält $CONTENT^{news \times string}$ noch das Element:
< N_{11}, "Boyer-Moore-Algorithmus für ...">
$CONN^{html \times news} = \Im(CONN^{html,news})$ =
{<$HTML_2$, N_7>, <$HTML_3$, N_8>, <$HTML_{18}$, N_7>, <$HTML_{18}$, N_8>}
$CONN^{news \times news} = \Im(CONN^{news,news})$ =
{<N_7, N_9>, <N_7, N_{10}>, <N_{10}, N_{11}>, <N_{11}, N_{12}>, <N_{12}, N_{13}>}
DESCRIPTION = $\Im(Description^{html,abstract})$ = {<$HTML_1$, $ABSTRACT_1$>}
IDENTIFIER = $\Im(Identifier^{unit,string})$ =
{<$HTML_1$, "http://www.cs.tu-berlin.de/~josefw/phd/index.html">,
<$HTML_2$, "http://www.cs.tu-berlin.de/~josefw/phd/introduction.html ">,
<$HTML_3$, "http://www.cs.tu-berlin.de/~josefw/phd/state.html">,
<$HTML_4$, "http://www.cs.tu-berlin.de/~josefw/phd/query.html">,
<$HTML_5$, "http://www.cs.tu-berlin.de/~josefw/phd/hynternetQL.html">,
<$HTML_6$, "http://www.cs.tu-berlin.de/~josefw/phd/literature.html">,
<$HTML_7$, "http://www.cs.tu-berlin.de/~josefw/phd/literature/Abiteboul,Beeri1995.html">,
<$HTML_8$, "http://www.cs.tu-berlin.de/~josefw/phd/literature/Ackermann,Hilb1972.htm">
<$HTML_9$, "http://www.cs.tu-berlin.de/~josefw/phd/literature/Afrati,Koutras1990.html">,
<$HTML_{10}$, "http://www.cs.huji.ac.il/~beeri">,
<$HTML_{11}$, "http://www-rocq.inria.fr/~abitebou/pub/icdt97.semistructured.talk.ps">,
<$HTML_{12}$, "http://web.nexor.co.uk/aliweb/aliweb">,

<HTML$_{13}$, "http://www.altavista.digital.com">,
<HTML$_{14}$, "http://apollo.co.uk">,
<HTML$_{15}$, "http://www.agentware.com">,
<HTML$_{16}$, "http://www.bigfoot.com">,
<HTML$_{17}$, "http://www.bunyip.com">,
<HTML$_{18}$, " http://www.javasoft.com">,
<ABSTRACT$_{1}$, " http://www.cs.tu-berlin.de/~josefw/phd/abs.abstract">}

Frage 1: Welche Abstracts hat das Dokument $html_1$?
$\Im((\lambda\ x^{abstract})\ Description^{a1,y1}(html_1, x^{abstract})) =$
$\{\langle U\rangle: \Im^{U}_{x}$ ist Modell von $Description^{html,abstract}(html_1, x^{abstract})$ für $U \in \mathcal{U}^{abstract}\} =$
$\{\langle U\rangle: \langle \Im^{U}_{x}(html_1), \Im^{U}_{x}(x^{abstract})\rangle \in \Im^{U}_{x}(Description^{html,abstract})$ für $U \in \mathcal{U}^{abstract}\} =$
$\{\langle U\rangle: \langle HTML_1, U\rangle \in$ DESCRIPTION für $U \in \mathcal{U}^{abstract}\} =$
$\{\langle ABSTRACT_1\rangle\}$

Frage 2: Welche Nachfolger der Stufe 1 bezüglich der Beziehungsart Conn haben die Teile der 3. Stufe des Dokuments $html_1$?
$\Im((\lambda\ x_\beta)\ \nabla_3(html_1, x_\alpha) \wedge \downarrow^{Conn}_{1}(x_\alpha, x_\beta) =$
$\{\langle U_\beta\rangle$: $\Im^{U\alpha U\beta}_{x\alpha\ x\beta}$ ist Modell von $\nabla_3(html_1, x_\alpha)$ und $\Im^{U\alpha U\beta}_{x\alpha\ x\beta}$ ist Modell von $\downarrow^{Conn}_{1}(x_\alpha, x_\beta)$
für $U_\alpha, U_\beta \in \mathcal{U}\}=$
$\{\langle U_\beta\rangle$: $\Im^{U\alpha U\beta U\gamma}_{x\alpha\ x\beta x\gamma}$ ist Modell von $(\exists f_1, \exists x_1, ..., \exists x_n)\ \nabla_2(html_1, f_1(x_1, ..., x_\alpha, ... , x_n))$ und

$\Im^{U\alpha U\beta U\gamma}_{x\alpha\ x\beta x\gamma}$ ist Modell von $Path^{Conn}(x_\alpha, x_\gamma, x_\beta)$ für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}=$
$\{\langle U_\beta\rangle$: $\Im^{U\alpha\ U\beta\ U\gamma\ U1...Un\ FU1}_{x\alpha\ x\beta\ x\gamma\ x1...xn\ f1}$ ist Modell von $\nabla_2(html_1, f_1(x_1, ..., x_\alpha, ... , x_n))$ und
$\Im^{U\alpha\ U\beta\ U\gamma\ U1...Un\ FU1}_{x\alpha\ x\beta\ x\gamma\ x1...xn\ f1}$ ist Modell von $Conn(x_\alpha, x_\gamma)$ und
$\Im^{U\alpha\ U\beta\ U\gamma\ U1...Un\ FU1}_{x\alpha\ x\beta\ x\gamma\ x1...xn\ f1}$ ist Modell von $Conn(x_\gamma, x_\beta)$ und
$\Im^{U\alpha\ U\beta\ U\gamma\ U1...Un\ FU1}_{x\alpha\ x\beta\ x\gamma\ x1...xn\ f1}(x_\alpha) \neq \Im^{U\alpha\ U\beta\ U\gamma\ U1...Un\ FU1}_{x\alpha\ x\beta\ x\gamma\ x1...xn\ f1}(x_\beta) \neq$
$\Im^{U\alpha\ U\beta\ U\gamma\ U1...Un\ FU1}_{x\alpha\ x\beta\ x\gamma\ x1...xn\ f1}(x_\gamma)$ und für mind. ein $FU_1 \in \{\rightarrow\}$
und für mind. ein $U_1, ..., U_n \in \mathcal{U}$ und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}=$
Bemerkung: Aus Übersichtlichkeitsgründen wird im Folgenden bei den Interpretationssymbolen die Interpretationsumgebung weggelassen (hier: $\Im^{U\alpha\ U\beta\ U\gamma\ U1...Un\ FU1}_{x\alpha\ x\beta\ x\gamma\ x1...xn\ f1}$)
$\{\langle U_\beta\rangle$: $\Im$ ist Modell von $(\exists f_2, \exists\ x_{21}, ..., \exists x_{2m})$
$\nabla_1(html_1, f_2(x_{21}, ..., f_1(x_1, ..., x_\alpha, ... , x_n), ..., x_{2m}))$ und
$\langle\Im(x_\alpha), \Im(x_\gamma)\rangle \in \Im(Conn)$ und $\langle\Im(x_\gamma), \Im(x_\beta)\rangle \in \Im(Conn)$ und $\Im(x_\alpha) \neq \Im(x_\beta) \neq \Im(x_\gamma)$
für mind. ein $FU_1 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n \in \mathcal{U}$ und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}=$
$\{\langle U_\beta\rangle$: $\Im$ ist Modell von $\nabla_1(html_1, f_2(x_{21}, ..., f_1(x_1, ..., x_\alpha, ... , x_n), ..., x_{2m}))$ und
$\langle U_\alpha, U_\gamma\rangle \in$ CONN und $\langle U_\gamma, U_\beta\rangle \in$ CONN $\Im(x_\alpha) \neq \Im(x_\beta) \neq \Im(x_\gamma)$
und für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathcal{U}$
und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}=$
$\{\langle U_\beta\rangle$: $\Im$ ist Modell von $(\exists f_3, \exists\ x_{31}, ..., \exists x_{3l})$
$Equal(html_1, f_3(x_{31}, ..., f_2(x_{21}, ..., f_1(x_1, ..., x_\alpha, ... , x_n), ..., x_{2m}), ..., x_{2l}))$
und $\langle U_\alpha, U_\gamma\rangle \in$ CONN und $\langle U_\gamma, U_\beta\rangle \in$ CONN und $U_\alpha \neq U_\beta \neq U_\gamma$
und für mind. ein $FU_1, FU_2 \in \{\rightarrow\}$ und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m} \in \mathcal{U}$
und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}=$
$\{\langle U_\beta\rangle$: $\Im$ ist Modell von

$Equal(html_1, f_3(x_{31}, ..., f_2(x_{21}, ..., f_1(x_1, ..., x_\alpha, ... , x_n), ..., x_{2m}), ..., x_{2l})))$ und
$<U_\alpha, U_\gamma> \in$ CONN und $<U_\gamma, U_\beta> \in$ CONN und $U_\alpha \neq U_\beta \neq U_\gamma$
und für mind. ein $FU_1, FU_2, FU_3 \in \{\rightarrow\}$
und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m}\ U_{31}, ..., U_{2l} \in \mathcal{U}$ und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}$=
$\{<U_\beta>$: $\mathfrak{I}(html_1) = \mathfrak{I}(f_3(x_{31}, ..., f_2(x_{21}, ..., f_1(x_1, ..., x_\alpha, ... , x_n), ..., x_{2m}), ..., x_{2l}))$ und
$<U_\alpha, U_\gamma> \in$ CONN und $<U_\gamma, U_\beta> \in$ CONN und $U_\alpha \neq U_\beta \neq U_\gamma$
und für mind. ein $FU_1, FU_2, FU_3 \in \{\rightarrow\}$ und
für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m}\ U_{31}, ..., U_{2l} \in \mathcal{U}$ und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}$=
$\{<U_\beta>$: $HTML_1$ =
$\mathfrak{I}(f_3)(\mathfrak{I}(x_{31}), ..., \mathfrak{I}(f_2)(\mathfrak{I}(x_{21}), ..., \mathfrak{I}(f_1)(\mathfrak{I}(x_1), ..., \mathfrak{I}(x_\alpha), ... ,\mathfrak{I}(x_n)), ..., \mathfrak{I}(x_{2m})) , ..., \mathfrak{I}(x_{2l})))$

und $<U_\alpha, U_\gamma> \in$ CONN und $<U_\gamma, U_\beta> \in$ CONN und $U_\alpha \neq U_\beta \neq U_\gamma$
und für mind. ein $FU_1, FU_2, FU_3 \in \{\rightarrow\}$
und für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m}\ U_{31}, ..., U_{2l} \in \mathcal{U}$ und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}$=
$\{<U_\beta>$: $HTML_1 = \rightarrow(U_{31}, ..., \rightarrow(U_{21}, ..., \rightarrow(U_1, ..., U_\alpha, ... , U_n), ..., U_{2m}) , ..., U_{3l}))$ und
$<U_\alpha, U_\gamma> \in$ CONN und $<U_\gamma, U_\beta> \in$ CONN und $U_\alpha \neq U_\beta \neq U_\gamma$ und
für mind. ein $U_1, ..., U_n, U_{21}, ..., U_{2m}\ U_{31}, ..., U_{2l} \in \mathcal{U}$ und für $U_\alpha, U_\beta, U_\gamma \in \mathcal{U}\}$=
$\{<N_7>, <N_8>\}$

Frage 3: Welche Dateien haben den Dateinamen "test.txt" und die Benutzerkennung "erhard@cs.tu-berlin.de" und liegen unterhalb des Dateiverzeichnisses "ftp.cs.tu-berlin.de/pub" bis zur Stufe 3 ?
$\mathfrak{I}((\lambda\ x^{file}_\alpha)\ \exists\ x^{file}_\beta$ File-name$^{file,string}(x^{file}_\alpha$ "test.txt") $\wedge$
User-id$^{file,string}(x^{file}_\alpha$, "erhard@cs.tu-berlin.de") $\wedge$
File-name$^{file,string}(x^{file}_\beta$, "ftp.cs.tu-berlin.de/pub") $\wedge$
$\Downarrow^{Contains}_3(x^{file}_\beta, x^{file}_\alpha))$ =
$\{<U_\alpha>$: $\mathfrak{I}^{U\alpha}_{x\alpha}$ ist Modell von $\exists\ x^{file}_\beta$ File-name$^{file,string}(x^{file}_\alpha$,"test.txt") $\wedge$
User-id$^{file,string}(x^{file}_\alpha$,"erhard@cs.tu-berlin.de") $\wedge$
File-name$^{file,string}(x^{file}_\beta$, "ftp.cs.tu-berlin.de/pub") $\wedge$
$\Downarrow^{Contains}_3(x^{file}_\beta, x^{file}_\alpha)$ für $U_\alpha \in \mathcal{U}^{file}\}$ =
$\{<U_\alpha>$: $\mathfrak{I}^{U\alpha\ U\beta}_{x\alpha\ x\beta}$ ist Modell von File-name$^{file,string}(x^{file}_\alpha$, "test.txt") und
$\mathfrak{I}^{U\alpha\ U\beta}_{x\alpha\ x\beta}$ ist Modell von User-id$^{file,string}(x^{file}_\alpha$, "erhard@cs.tu-berlin.de") und
$\mathfrak{I}^{U\alpha\ U\beta}_{x\alpha\ x\beta}$ ist Modell von File-name$^{file,string}(x^{file}_\beta$, "ftp.cs.tu-berlin.de/pub") und
$\mathfrak{I}^{U\alpha\ U\beta}_{x\alpha\ x\beta}$ ist Modell von $\Downarrow^{Contains}_3(x^{file}_\beta, x^{file}_\alpha)$ für $U_\alpha \in \mathcal{U}^{file}$ und
für mindestens ein $U_\beta \in \mathcal{U}^{file}\}$ =
$\{<U_\alpha>$: $<\mathfrak{I}^{U\alpha}_{x\alpha}(x^{file}_\alpha)$, $\mathfrak{I}$("test.txt")> $\in \mathfrak{I}$(File-name^{file,string}) und
$<\mathfrak{I}^{U\alpha}_{x\alpha}(x^{file}_\alpha)$, $\mathfrak{I}$("erhard@cs.tu-berlin.de")> $\in \mathfrak{I}$(User-id^{file,string}) und
$<\mathfrak{I}^{U\beta}_{x\beta}(x^{file}_\beta)$, $\mathfrak{I}$("ftp.cs.tu-berlin.de/pub")> $\in \mathfrak{I}$(File-name^{file,string}) und
($\mathfrak{I}^{U\alpha\ U\beta}_{x\alpha\ x\beta}$ ist Modell von $\downarrow^{Contains}_3(x^{file}_\beta, x^{file}_\alpha)$ oder
$\mathfrak{I}^{U\alpha\ U\beta}_{x\alpha\ x\beta}$ ist Modell von $\downarrow^{Contains}_2(x^{file}_\beta, x^{file}_\alpha)$ oder
$\mathfrak{I}^{U\alpha\ U\beta}_{x\alpha\ x\beta}$ ist Modell von $\downarrow^{Contains}_1(x^{file}_\beta, x^{file}_\alpha))$ für $U_\alpha \in \mathcal{U}^{file}$ und
für mindestens ein $U_\beta \in \mathcal{U}^{file}\}$ =
$\{<U_\alpha>$: $<U_\alpha$, "test.txt"> $\in$ FILENAME und
$<U_\alpha$, "erhard@cs.tu-berlin.de"> $\in$ USERID und
$< U_\beta$, "ftp.cs.tu-berlin.de/pub"> $\in$ FILENAME und
($\mathfrak{I}^{U\alpha\ U\beta\ U1\ U2\ U3}_{x\alpha\ x\beta\ x1\ x2\ x3}$ ist Modell von Path$^{Contains}(x^{file}_\beta, x^{file}_1, x^{file}_2, x^{file}_3, x^{file}_\alpha)$ oder
$\mathfrak{I}^{U\alpha\ U\beta\ U1\ U2}_{x\alpha\ x\beta\ x1\ x2}$ ist Modell von Path$^{Contains}(x^{file}_\beta, x^{file}_1, x^{file}_2, x^{file}_\alpha)$ oder

$\Im^{U\alpha\, U\beta\, U1}{}_{x\alpha\, x\beta\, x1}$ ist Modell von $Path^{Contains}(x^{file}{}_{\beta}, x^{file}{}_{1}, x^{file}{}_{\alpha}))$ für $U_\alpha \in \mathit{U}^{file}$ und
für mindestens ein $U_\beta, U_1, U_2, U_3 \in \mathit{U}^{file}\} =$

Bemerkung: Aus Übersichtlichkeitsgründen wird im Folgenden jeweils bei den Interpretationssymbolen die Interpretationsumgebung weggelassen (hier: $\Im^{U\alpha\, U\beta\, U1}{}_{x\alpha\, x\beta\, x1}$)

$\{<U_\alpha>: <U_\alpha$, "test.txt"$> \in$ FILENAME und
$<U_\alpha$, "erhard@cs.tu-berlin.de"$> \in$ USERID und
$< U_\beta$, "ftp.cs.tu-berlin.de/pub"$> \in$ FILENAME und
$((\Im$ ist Modell von $Contains(x_\beta, x_1)$ und $\Im$ ist Modell von $Contains(x_1, x_2)$
und $\Im$ ist Modell von $Contains(x_2, x_3)$ und $\Im$ ist Modell von $Contains(x_3, x_\alpha)$ und
$\Im(x_\alpha) \neq \Im(x_\beta) \neq \Im(x_1) \neq \Im(x_2) \neq \Im(x_3))$ oder
$(\Im$ ist Modell von $Contains(x_\beta, x_1)$ und $\Im$ ist Modell von $Contains(x_1, x_2)$
und $\Im$ ist Modell von $Contains(x_2, x_\alpha)$ und $\Im(x_\alpha) \neq \Im(x_\beta) \neq \Im(x_1) \neq \Im(x_2))$ oder
$(\Im$ ist Modell von $Contains(x_\beta, x_1)$ und $\Im$ ist Modell von $Contains(x_1, x_\alpha)$ und
$\Im(x_\alpha) \neq \Im(x_\beta) \neq \Im(x_1)))$ für $U_\alpha \in \mathit{U}^{file}$ und für mindestens ein $U_\beta, U_1, U_2, U_3 \in \mathit{U}^{file}\} =$

$\{<U_\alpha>: <U_\alpha$, "test.txt"$> \in$ FILENAME und
$<U_\alpha$, "erhard@cs.tu-berlin.de"$> \in$ USERID und
$<U_\beta$, "ftp.cs.tu-berlin.de/pub"$> \in$ FILENAME und
$(<\Im(x_\beta), \Im(x_1)> \in \Im(Contains)$ und $<\Im(x_1), \Im(x_2)> \in \Im(Contains)$
und $< \Im(x_2), \Im(x_3)> \in \Im(Contains)$ und $<\Im(x_3), \Im(x_\alpha)> \in \Im(Contains)$ und
$U_\alpha \neq U_\beta \neq U_1 \neq U_2 \neq U_3)$ oder
$(<\Im(x_\beta), \Im(x_1)> \in \Im(Contains)$ und $<\Im(x_1), \Im(x_2)> \in \Im(Contains)$
und $<\Im(x_2), \Im(x_\alpha)> \in \Im(Contains)$ und $U_\alpha \neq U_\beta \neq U_1 \neq U_2)$ oder
$(<\Im(x_\beta), \Im(x_1)> \in \Im(Contains)$ und $<\Im(x_1), \Im(x_\alpha)> \in \Im(Contains)$ und $U_\alpha \neq U_\beta \neq U_1))$
für $U_\alpha \in \mathit{U}^{file}$ und für mindestens ein $U_\beta, U_1, U_2, U_3 \in \mathit{U}^{file}\} =$

$\{<U_\alpha>: <U_\alpha$, "test.txt"$> \in$ FILENAME und
$<U_\alpha$, "erhard@cs.tu-berlin.de"$> \in$ USERID und
$< U_\beta$, "ftp.cs.tu-berlin.de/pub"$> \in$ FILENAME und
$((<U_\beta, U_1> \in$ CONTAINS und $<U_1, U_2> \in$ CONTAINS und $<U_2, U_3> \in$ CONTAINS
und $<U_3, U_\alpha> \in$ CONTAINS und $U_\alpha \neq U_\beta \neq U_1 \neq U_2 \neq U_3)$ oder
$(<U_\beta, U_1> \in$ CONTAINS und $<U_1, U_2> \in$ CONTAINS und
$<U_2, U_\alpha> \in$ CONTAINS und $U_\alpha \neq U_\beta \neq U_1 \neq U_2)$ oder
$(<U_\beta, U_1> \in$ CONTAINS und $<U_1, U_\alpha> \in$ CONTAINS und $U_\alpha \neq U_\beta \neq U_1))$
für $U_\alpha \in \mathit{U}^{file}$ und für mindestens ein $U_\beta, U_1, U_2, U_3 \in \mathit{U}^{file}\} =$

$\{<F_6>, <F_7>\}$

Frage 4: Welche Artikel liegen in der Newsgroup "comp.lang.java.programmer" oder sind Nachfolger bis zur Stufe 3 des Dokuments, deren URL "http://www.javasoft.com" ist und enthalten den regulären Ausdruck "String*arbeitung" in ihrem "Content" ?

$\Im((\lambda\ x^{news}{}_{\alpha})\ \exists\ x^{unit}{}_{\beta}\ \exists\ x^{string}{}_{\gamma}\ Newsgroup^{news,string}(x^{news}{}_{\alpha}$, "comp.lang.java.programmer"$) \vee$
$(Identifier^{unit,string}(x^{unit}{}_{\beta}$, " http://www.javasoft.com"$) \wedge \Downarrow^{P}{}_{3}(x^{unit}{}_{\beta}, x^{news}{}_{\alpha})) \wedge$
$Content(x^{news}{}_{\alpha}, x^{string}{}_{\gamma}) \wedge Contains($"String*arbeitung"$, x^{string}{}_{\gamma}) =$

$\{<U_\alpha>: \Im^{U\alpha}{}_{x\alpha}$ ist Modell von
$\exists\ x^{unit}{}_{\beta}\ \ \exists\ x^{string}{}_{\gamma}\ Newsgroup^{news,string}(x^{news}{}_{\alpha}$, "comp.lang.java.programmer"$) \vee$
$(Identifier^{unit,string}(x^{unit}{}_{\beta}$, " http://www.javasoft.com"$) \wedge \Downarrow^{P}{}_{3}(x^{unit}{}_{\beta}, x^{news}{}_{\alpha})) \wedge$
$Content(x^{news}{}_{\alpha}, x^{string}{}_{\gamma}) \wedge Contains($"String*arbeitung"$, x^{string}{}_{\gamma})$ für $U_\alpha \in \mathit{U}^{news}\} =$

$\{<U_\alpha>: \Im^{U\alpha\, U\beta\, U\gamma}{}_{x\alpha\, x\beta\, x\gamma}$ ist Modell von
$Newsgroup^{news,string}(x^{news}{}_{\alpha}$, "comp.lang.java.programmer"$) \vee$

$(\text{Identifier}^{unit,string}(x^{unit}_{\beta}, \text{"http://www.javasoft.com"}) \wedge$
$\Downarrow^{P}_{3}(x^{unit}_{\beta}, x^{news}_{\alpha})) \wedge \text{Content}(x^{news}_{\alpha}, x^{string}_{\gamma}) \wedge \text{Contains}(\text{"String*arbeitung"}, x^{string}_{\gamma})$
für $U_{\alpha} \in U^{news}$ und für mindestens ein $U_{\beta} \in U^{unit}$ und für mindestens ein $U_{\gamma} \in U^{string}$ } =
$\{<U_{\alpha}>$:
$<\Im^{U\alpha\;U\beta\;U\gamma}_{x\alpha\;x\beta\;x\gamma}(x^{news}_{\alpha}),\ \Im^{U\alpha\;U\beta\;U\gamma}_{x\alpha\;x\beta\;x\gamma}(\text{"comp.lang.java.programmer"})>$
$\in$ NEWSGROUP
oder $(<\Im^{U\alpha\;U\beta\;U\gamma}_{x\alpha\;x\beta\;x\gamma}(x^{unit}_{\beta}),\ \Im^{U\alpha\;U\beta\;U\gamma}_{x\alpha\;x\beta\;x\gamma}(\text{"http://www.javasoft.com"})>$
$\in$ IDENTIFIER
und ($\Im^{U\alpha\,U\beta\,U\gamma}_{x\alpha\,x\beta\,x\gamma}$ ist Modell von $\downarrow^{P}_{3}(x^{unit}_{\beta}, x^{news}_{\alpha})$
oder $\Im^{U\alpha\,U\beta\,U\gamma}_{x\alpha\,x\beta\,x\gamma}$ ist Modell von $\downarrow^{P}_{2}(x^{unit}_{\beta}, x^{news}_{\alpha})$
oder $\Im^{U\alpha\,U\beta\,U\gamma}_{x\alpha\,x\beta\,x\gamma}$ ist Modell von $\downarrow^{P}_{1}(x^{unit}_{\beta}, x^{news}_{\alpha})$))
und $<\Im^{U\alpha\,U\beta\,U\gamma}_{x\alpha\,x\beta\,x\gamma}(x^{news}_{\alpha}), \Im^{U\alpha\,U\beta\,U\gamma}_{x\alpha\,x\beta\,x\gamma}(x^{string}_{\gamma})> \in$ CONTENT
und $\Im^{U\alpha\,U\beta\,U\gamma}_{x\alpha\,x\beta\,x\gamma}$ ist Modell von Contains("String*arbeitung", x^{string}_{γ})
für $U_{\alpha} \in U^{news}$ und für mindestens ein $U_{\beta} \in U^{unit}$ und für mindestens ein $U_{\gamma} \in U^{string}$ } =
$\{<U_{\alpha}>: <U_{\alpha}, \text{"comp.lang.java.programmer"}> \in$ NEWSGROUP oder
$(<U_{\beta}, \text{"http://www.javasoft.com"}> \in$ IDENTIFIER und
($\Im^{U\alpha\,U\beta\,U\gamma\,U1\,U2\,U3}_{x\alpha\,x\beta\,x\gamma\;x1\,x2\,x3}$ ist Modell von $\text{Path}^{P}(x^{unit}_{\beta}, x_1, x_2, x_3, x^{news}_{\alpha})$
oder $\Im^{U\alpha\,U\beta\,U\gamma\,U1\,U2\,U3}_{x\alpha\,x\beta\,x\gamma\;x1\,x2\,x3}$ ist Modell von $\text{Path}^{P}(x^{unit}_{\beta}, x_1, x_2, x^{news}_{\alpha})$
oder $\Im^{U\alpha\,U\beta\,U\gamma\,U1\,U2\,U3}_{x\alpha\,x\beta\,x\gamma\,x1\,x2\,x3}$ ist Modell von $\text{Path}^{P}(x^{unit}_{\beta}, x_1, x^{news}_{\alpha})$))
und $< U_{\alpha}, U_{\gamma}> \in$ CONTENT und U_{γ} enthält den regulären Ausdruck "String*arbeitung"
für $U_{\alpha} \in U^{news}$ und für mindestens ein $U_{\beta} \in U^{unit}$ und für mindestens ein $U_{\gamma} \in U^{string}$
und für mindestens ein $U_1, U_2, U_3 \in U$ } =
Bemerkung: wir lassen aus Übersichtlichkeitsgründen im Folgenden jeweils bei den Interpretationssymbolen die Interpretationsumgebung weg (hier: $\Im^{U\alpha\,U\beta\,U\gamma\,U1\,U2\,U3}_{x\alpha\,x\beta\,x\gamma\,x1\,x2\,x3}$)
$\{<U_{\alpha}>: < U_{\alpha}, \text{"comp.lang.java.programmer"}> \in$ NEWSGROUP oder
$(<U_{\beta}, \text{"http://www.javasoft.com"}> \in$ IDENTIFIER und
($\Im$ ist Modell von $P(x^{unit}_{\beta}, x_1)$ und $\Im$ ist Modell von $P(x^{unit}_{1}, x_2)$ und
$\Im$ ist Modell von $P(x^{unit}_{2}, x_3)$ und $\Im$ ist Modell von $P(x_3, x^{news}_{\alpha})$ und
$\Im(x^{news}_{\alpha}) \neq \Im(x^{unit}_{\beta}) \neq \Im(x_1) \neq \Im(x_2) \neq \Im(x_3)$) oder
($\Im$ ist Modell von $P(x^{unit}_{\beta}, x_1)$ und $\Im$ ist Modell von $P(x^{unit}_{1}, x_2)$ und
$\Im$ ist Modell von $P(x_2, x^{news}_{\alpha})$ und $\Im(x^{news}_{\alpha}) \neq \Im(x^{unit}_{\beta}) \neq \Im(x_1) \neq \Im(x_2)$) oder
($\Im$ ist Modell von $P(x^{unit}_{\beta}, x_1)$ und $\Im$ ist Modell von $P(x_1, x^{news}_{\alpha})$ und
$\Im(x^{news}_{\alpha}) \neq \Im(x^{unit}_{\beta}) \neq \Im(x_1)$))))
und $< U_{\alpha}, U_{\gamma}> \in$ CONTENT und U_{γ} enthält den regulären Ausdruck "String*arbeitung"
für $U_{\alpha} \in U^{news}$ und für mindestens ein $U_{\beta} \in U^{unit}$ und für mindestens ein $U_{\gamma} \in U^{string}$
und für mindestens ein $U_1, U_2, U_3 \in U$ } =
$\{<U_{\alpha}>: <U_{\alpha}, \text{"comp.lang.java.programmer"}> \in$ NEWSGROUP oder
$(<U_{\beta}, \text{"http://www.javasoft.com"}> \in$ IDENTIFIER und
$(<\Im(x^{unit}_{\beta}), \Im(x_1)> \in \Im(P)$ und $<\Im(x_1), \Im(x_2)> \in \Im(P)$ und
$<\Im(x_2), \Im(x_3)> \in \Im(P)$ und $<\Im(x_3), \Im(x_{\alpha})> \in \Im(P)$ und $U_{\alpha} \neq U_{\beta} \neq U_1 \neq U_2 \neq U_3$) oder
$(<\Im(x^{unit}_{\beta}), \Im(x_1)> \in \Im(P)$ und $<\Im(x_1), \Im(x_2)> \in \Im(P)$ und
$<\Im(x_2), \Im(x_{\alpha})> \in \Im(P)$ und $U_{\alpha} \neq U_{\beta} \neq U_1 \neq U_2$) oder
$(<\Im(x^{unit}_{\beta}), \Im(x_1)> \in \Im(P)$ und $<\Im(x_1), \Im(x_{\alpha})> \in \Im(P)$ und $U_{\alpha} \neq U_{\beta} \neq U_1$)))
und $< U_{\alpha}, U_{\gamma}> \in$ CONTENT und U_{γ} enthält den regulären Ausdruck "String*arbeitung"
für $U_{\alpha} \in U^{news}$ und für mindestens ein $U_{\beta} \in U^{unit}$ und für mindestens ein $U_{\gamma} \in U^{string}$
und für mindestens ein $U_1, U_2, U_3 \in U$ } =
$\{<U_{\alpha}>: <U_{\alpha}, \text{"comp.lang.java.programmer"}> \in$ NEWSGROUP oder

(<U_β, "http://www.javasoft.com"> $\in$ IDENTIFIER und
(<U_β, U_1> $\in U \times U$ und <U_1, U_2> $\in U \times U$ und <U_2, U_3> $\in U \times U$ und
<U_3, U_α> $\in U \times U$ *und* $U_\alpha \neq U_\beta \neq U_1 \neq U_2 \neq U_3$) oder
(<U_β, U_1> $\in U \times U$ und <U_1, U_2> $\in U \times U$ und <U_2, U_α> $\in U \times U$ und $U_\alpha \neq U_\beta \neq U_1 \neq U_2$)
oder (<U_β, U_1> $\in U \times U$ und <U_1, U_α> $\in U \times U$ *und* $U_\alpha \neq U_\beta \neq U_1$)))
und < U_α, U_γ> $\in$ CONTENT und U_γ enthält den regulären Ausdruck "String*arbeitung"
für $U_\alpha \in U^{news}$ und für mindestens ein $U_\beta \in U^{unit}$ und für mindestens ein $U_\gamma \in U^{string}$
und für mindestens ein U_1, U_2, $U_3 \in U$ } =
{<N_3>, <N_4>, <N_{11}>}

4.6 Vergleich mit anderen Anfragesprachen

Die Anfragesprache StructuredQL wird im Folgenden mit anderen logischen Anfragesprachen verglichen: mit dem Beeri-Kornatzky-Modell (Beeri, Kornatzky (1990)), mit IQL (Reiner (1991)) und mit dem DATALOG-Modell von Fuhr (Fuhr (1995)). Jede Anfragesprache wird kurz vorgestellt und anhand von Vergleichskriterien eingeordnet. Eine genaue Klassifikation z.B. nach Anfragemächtigkeit bleibt der weiteren Forschung vorbehalten.

4.6.1 Beeri-Kornatzky-Modell

Beeri, Kornatzky bauen eine logische Anfragesprache auf, deren Hauptziel darin liegt, die Netzstruktur von Hypertexten abzubilden. Anfragen mittels netzspezifischer Angaben ergeben wiederum Hypertextnetzwerke, so daß geschachtelte Anfragen wie in SQL möglich sind. Anfragen werden ausgehend von ein oder mehreren Dokumenten gestellt. Der besondere Schwerpunkt bei Beeri, Kornatzky liegt in der Verwendung von Modaloperatoren. Für die Sprache wird die Syntax vollständig und die Semantik zum Teil festgelegt. Beispiele werden vorgestellt.

Art der Logik	Mischung von Propositionallogik (ohne Prädikate und Variablen) und Modallogik (nach eigenen Angaben)
Suche mit Attributen	Attribute von Dokumenten und Attribute von Beziehungen (z.B. mit Benennung und Gewichtung)
Vergleichsoperatoren	=, ≠
Volltextsuche	nicht vorhanden
Quantoren	∃ (es existiert ein), ∀ (für alle), at least n, exactly n, few, many, most
Boolesche Verknüpfungen	¬ (nicht), ∨ (oder)
Netzwerkoperatoren	Beziehung, Pfad mit Richtungsangaben: forward, backward, undirected
geschachtelte Dokumente	nicht vorhanden
Suchobjekte	what (entspricht dem λ-Operator): Dokumente (Knoten) und Beziehungen
Syntax, Semantik	Syntax explizit und vollständig, Semantik unvollständig; zum Teil anhand von Beispielen erklärt
Beispiele	Ergebnis der Anfrage wird gegeben; einzelne Auswertungsschritte fehlen
Besonderheiten	Benutzerorientierte Anfragen sind möglich: Definition von

	"Views", Ähnlichkeitsfunktionen aus traditionellen Informationssystemen können durch modale Operatoren (Quantoren) eingebracht werden. Als Mengenoperatoren werden die Vereinigung und die Differenz zur Verfügung gestellt. Bei den Netzwerkoperatoren kann keine Suchtiefe spezifiziert werden. Weiterhin fehlen Operatoren für Stationen und Geschwister.

4.6.2 IQL

Reiner baut eine prädikatenlogische Anfragesprache zur Suche nach Dokumenten, Fakten und Erklärungen auf. Syntax und Semantik werden streng getrennt, zahlreiche Anfragebeispiele mit den einzelnen Auswertungsschritten werden gegeben. Ein besonderer Schwerpunkt liegt bei ihr in dem Aufbau einer Anfragesprache für die Volltextsuche (Reiner (1991), Reiner (1994)). Link-strukturierte und geschachtelte Dokumente wurden von ihr bislang nicht untersucht.

Art der Logik	Prädikatenlogik 1. Stufe
Suche mit Attributen	Attribute von Dokumenten
Vergleichsoperatoren	=, ≠, <, >, ≤, ≥
Volltextsuche	universelle Anfragesprache (siehe Reiner (1994))
Quantoren	∃ (es existiert ein), ∀ (für alle)
Boolesche Verknüpfungen	¬ (nicht), ∨ (oder), ∧ (und)
Netzwerkfunktionen	für Thesauri (Bäume): Vorfahren (mit Suchtiefe), Nachfahren (mit Stufe), Geschwister, Synonyme
Geschachtelte Dokumente	nicht vorhanden
Suchobjekte	λ-Operator für Dokumente (Dokumente), Fakten, Erklärungen
Syntax, Semantik	explizit und vollständig
Beispiele	Auswertungsschritte werden für jede Teilsprache anhand von Beispielmodellen detailiert vorgeführt
Besonderheiten	benutzerorientierte Anfragesprachen (UUQL, TUQL), alle Sprachen wurden implementiert, die Autorin erhielt für ihre Arbeit den Erich-Pietsch-Preis der Deutschen Gesellschaft für Dokumentation

4.6.3 Datalog-Modell von Fuhr

Fuhr stellt in der Sprache Datalog gebildete Programme vor, die Anfragemöglichkeiten für link-strukturierte und geschachtelte Dokumente modellieren. Dokumente werden durch Beziehungsprädikate miteinander verbunden und Teilobjekte durch Aggregationsprädikate gebildet. Es ist beispielsweise möglich, SGML-Elemente in Datalog nachzubilden. Das HTML-Element ul(li(a),li(b)) würde z.B. durch folgende Datalog-Fakten repräsentiert:
root(ul1).
ul(ul1, ul1li1).
ul(ul1, ul1li2).
li(ul1li1, a).
li(ul1li2, b).

Art der Logik	Variante der Hornklausel-Logik (Datalog)
Suche mit Attributen	Attribute von Dokumenten
Vergleichsoperatoren	Built-in-Prädikate: =, ≠, <, >, ≤, ≥
Volltextsuche	Nicht vorhanden
Quantoren	Der Allquantor ∀ wird implizit für alle Individuenvariablen verwendet
Boolesche Verknüpfungen	Es können Formeln der Form: $A \leftarrow B_1 \vee ... \vee B_n$ gebildet werden
Netzwerkoperatoren	Beziehung, Nachfahren
Geschachtelte Dokumente	Rekursive Strukturen werden durch Datalog-Regeln ausgedrückt, Teilstrukturen können mit Datalog-Regeln bestimmt werden
Suchobjekte	Dokumente (Dokumente, Bücher, Proceedings, Artikel etc.)
Syntax, Semantik	explizit und vollständig in Ceri, Gottlob, Tanca (1990)
Beispiele	einige Datalog-Programme
Besonderheiten	Verwendung des um die Negation erweiterten Datalog (stratified Datalog), Operatoren für terminologische Logiken: not, and, all, some, atleast, atmost, u.a., es können keine Funktionen ausgedrückt werden, Modellierung von Multimedia-Dokumenten, der Prototyp eines Datalog-Interpreters wurde am Lehrstuhl von Herrn Fuhr implementiert.

4.6.4 StructuredQL

StructuredQL ist eine prädikatenlogische Sprache 2. Stufe zur Suche in strukturierten Dokumenten. Syntax und Semantik werden streng getrennt. Für jede Teilsprache werden Anfragebeispiele mit den Auswertungsschritten gegeben. Ein besonderer Schwerpunkt liegt in der Abbildung von Netzwerkfunktionen und in der Behandlung geschachtelter Objekte. Weiterhin werden Sorten von Dokumenten (z.B. Artikel, Dateien etc.) und von Beziehungen unterschieden. Zur Volltextsuche werden verschieden Operatoren bereitgestellt.

Art der Logik	mehrstufige Prädikatenlogik
Suche mit Attributen	Attribute von Dokumenten
Vergleichsoperatoren	≠, <, >, ≤, ≥
Volltextsuche	reguläre Ausdrücke, Near-Operator, linguistische Operatoren, Thesaurusoperatoren
Quantoren	∃ (es existiert ein), ∀ (für alle)
Boolesche Verknüpfungen	¬ (nicht), ∨ (oder), ∧ (und), → (Implikation), ↔ (Äquivalenz)
Netzwerkoperatoren	Beziehung, Pfad, Vorgänger/Nachfolger (mit Stufe), Geschwister, Stationen
Geschachtelte Dokumente	Prädikat für Teile/Behälter, Teilobjekte werden durch Funktionen bestimmt, Sorten von Dokumenten und Beziehungen
Suchobjekte	λ-Operator für Dokumente (Artikel, Dateien, Dokumente/Dokumente, E-Mails, Menüs, Personen/Gruppen, Rechner), Deskriptoren, Beziehungen, Teile/Behälter von Objekten
Syntax, Semantik	explizit und vollständig
Beispiele	Auswertungsschritte werden für jede Teilsprache anhand von Beispielmodellen detailiert vorgeführt

5 Architektur eines universellen Internet-Informationssystems

Nach dem Aufbau von Anfragesprachen für Internet-Informationssysteme werden zwei Architekturen (dezentrale Architektur, zentrale/dezentrale Architektur) zur Realisierung eines universellen Internet-Informationssystems vorgestellt.
In beiden Architekturen werden eindeutig spezifizierte Sprachebenen voneinander abgetrennt. Dadurch ist es möglich, systemspezifischen Eigenheiten in der heterogenen Landschaft der Informationssysteme zu entgehen. Der Benutzer verwendet Standardbrowser (z.B. Netscape, Microsoft Explorer) oder andere Standard-Benutzeroberflächen (z.B. Java-Oberfläche, MS-Windows-Oberfläche) zur Kommunikation mit dem Suchsystem. Zur Effizienzsteigerung werden Anfragen und Anfrageergebnisse mit Hilfe des Cache Systems zwischengespeichert.
Zur Vereinfachung wird in beiden Architekturen die Schicht einer gesonderten Benutzeranfragesprache (zusätzlich zur universellen Anfragesprache) und auch das Session- und Benutzerprofil-Management weggelassen.
Auf die Vorstellung einer rein zentralen Architektur wird verzichtet, da dies dem dynamischen und dezentralen Charakter des Internet widerspricht.

5.1 Dezentrale Architektur

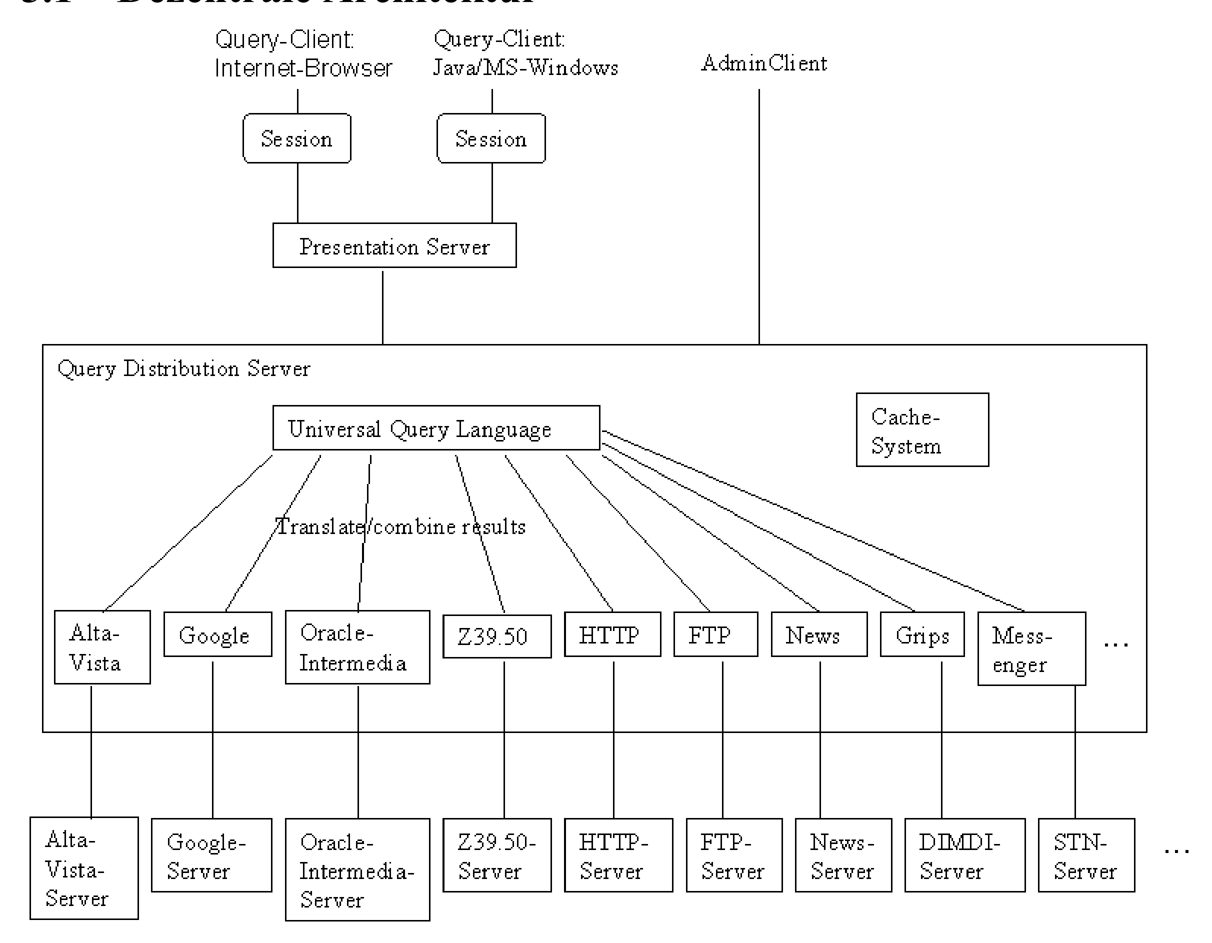


Abbildung 41: dezentrale Architektur

Das Suchsystem besteht aus den Komponenten Presentation-Server und Query Distribution-Server. Die Aufgabe des Presentation-Servers besteht darin, Anfragen von Benutzern entgegenzunehmen und zur Beantwortung an den Query Distribution Server zu senden. Dieser

übersetzt die universelle Anfrage mit den Sprachübersetzern in die systemabhängigen Anfragesprachen und sendet die Anfrage an die externen Internet-Informationsysteme weiter und gibt das aus den Systemen zusammengeführte Suchergebnis an den Client zurück.

Hauptvorteil der dezentralen Architektur ist deren einfache und rasche Realisierbarkeit. Die Entwicklung eines effizienten und mehrbenutzerfähigen Collection Managers entfällt. Die Anforderungen an die Hardware sind relativ niedrig, da keine zentrale Datenhaltung durchgeführt wird. Autoren verwalten Dokumente dezentral mit ihren gewohnten Editierwerkzeugen (XML-Edititor, MS-Word etc.). Zielsysteme mit ihren Beständen sind einfach anschließbar. Zwischen der universellen Anfragesprache und der zielsystemabhängigen Anfragesprache wird jeweils ein Sprachübersetzer bereitgestellt. Falls z.B. ein neuartiges Informationssystem entwickelt wird, braucht so nicht das Gesamtsystem modifiziert zu werden, sondern es wird der Übersetzer für dieses spezielle System neu eingesetzt.
Der Ausfall von Zielsystemen beeinträchtigt nicht den Gesamtbetrieb des Systems. Suchergebnisse sind mit einer hohen Wahrscheinlichkeit verfügbar. Durch redundante Anbindung mehrerer Zielsysteme mit gleichen Beständen kann die Verfügbarkeit der Bestände weiter erhöht werden.

Der Hauptnachteil der dezentralen Architektur liegt darin, daß die Mächtigkeit der universellen Anfragesprache nicht erreicht werden kann, da die Zielsysteme keine ausreichende und zudem eine unterschiedliche Anfragemächtigkeit aufweisen. Folgendes Beispiel soll dies illustrieren:
Gegeben sei der Dokumentenbestand D = {doc1, doc2, doc3, doc4}
Im Informationssystem IS1 seien folgende Suchergebnismengen mit den Anfragemöglichkeiten bildbar: A1 = {{doc1},{doc2},{doc3},{doc1, doc2},{doc1, doc3}}
Im Informationssystem IS2 seien folgende Suchergebnismengen mit den Anfragemöglichkeiten bildbar: A2 = {{doc1},{doc2},{doc3}, {doc1, doc2},{doc1, doc3}, {doc1, doc2, doc3}}
Im Informationssystem IS3 seien folgende Suchergebnismengen mit den Anfragemöglichkeiten bildbar: A3 = {{doc1},{doc2},{doc1, doc3},{doc1, doc2, doc3}, {doc4}}
Die universelle Anfragesprache kann dann nicht die Vereinigungsmenge sondern nur die Schnittmenge der Mengen A1, A2 und A3 = {{doc1},{doc2},{doc1, doc3}} nachbilden.

Die Weiterentwicklung der Zielsysteme hin zur Unterstützung der Suche in geschachtelten Dokumenten (XQL, Xpath, Xlink etc.) und zur Suche in link-strukturierten Dokumenten ist erst mittelfristig zu erwarten.
Ein weiterer Nachteil der dezentralen Architektur liegt in der inkonsistenten Zusammenführung von Suchergebnissen. Beispielsweise werden Dubletten in überlappenden Beständen in den Zielsystemen nicht korrekt erkannt oder es werden bestimmte Felder nicht angezeigt, da sie im Zielsystem nicht verfügbar sind. Weiterhin müssen große Treffermengen der Zielsysteme aus Effizienzgründen bei der Übermittlung an das Gesamtsystem beschränkt werden. Dadurch ist die Vollständigkeit des Suchergebnisses nicht mehr gewährleistet. Zudem kann das neuberechnete Ranking des Suchergebnisses dadurch verfälscht werden.
Die Anfrageperformance wird durch die dezentrale Architektur vermindert. Sie ist zudem von der Performance der Zielsysteme abhängig. Bestimmte Zielsysteme können überhaupt nicht angebunden werden, da ihre Performance zu gering ist. Durch die Realisierung eines skalierbaren Cache-Systems für Anfragen und Suchergebnisse kann die Performance jedoch stark verbessert werden.
Ein weiterer Nachteil der dezentralen Architektur liegt darin, daß der Gesamtbestand des Systems nur grob geschätzt werden kann, da die Teilbestände der Zielsysteme überlappen und

z.T. von unvorhersehbaren Faktoren wie z.B. Ausfall des Zielsystems, Umkonfiguration des Zielsystems etc. verfälscht werden. Die Aktualität des Gesamtbestands ist zudem abhängig von der Aktualität der Zielsystembestände.
Der Aufwand für Konfigurationsänderungen am System sollte nicht unterschätzt werden. Veränderungen der Schnittstellen zum Zielsystem treten häufig auf und haben jeweils zentrale Konfigurationsänderungen zur Folge.

Ein stark vereinfachter Prototyp (URL-Echo-Sounder), der eine kleine Teilmenge der beschriebenen Anfragesprache abbildet, wurde realisiert (siehe Anhang 10.5).

5.2 Zentral/Dezentrale Architektur

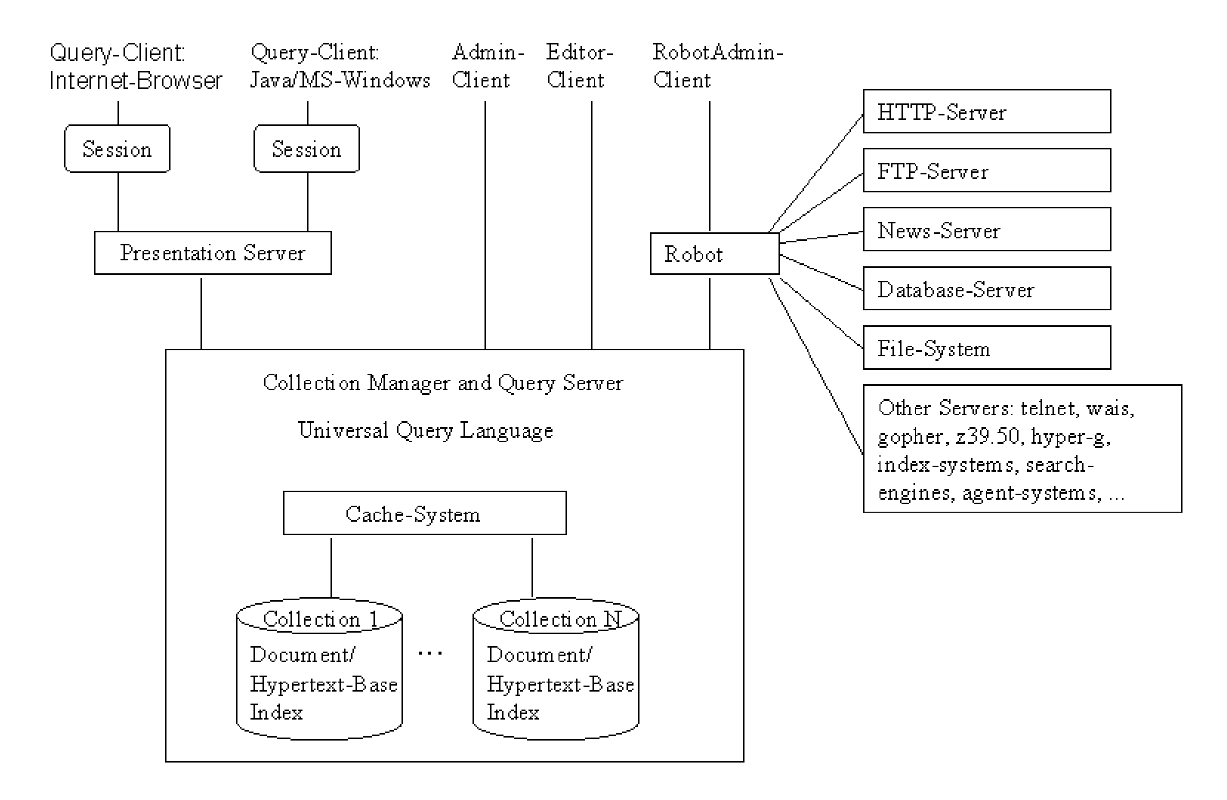


Abbildung 42: zentrale/dezentrale Architektur

Das Suchsystem besteht aus den Komponenten Presentation-Server, Collection Manager and Query Server und Robot. Die Aufgabe des Presentation-Servers besteht darin, Anfragen von Benutzern entgegenzunehmen und zur Beantwortung an den Collection Manager and Query Server weiterzuleiten und die Antwort in geigneter Form an den Client zurückzusenden (query managing). Die Bestände (Collections) werden direkt durch die Autoren oder mit Hilfe des Roboters aufgebaut und verwaltet. Roboter sammeln öffentlich zugängliche Datenbestände in bestimmten Zeitabständen auf und übergeben sie dem Collection Manager Server.

Hauptvorteil der zentral/dezentralen Architektur ist die Möglichkeit, die Mächtigkeit der universellen Anfragesprache performant abzubilden. Es sind keine Übersetzer zu weiteren Informationssystemen erforderlich. Es werden Bestände (Collections) unterschiedlichen Typs lokal bis global verwaltet. Autoren verwalten ihre Dokumente zentral oder dezentral mit gewohnten Editierwerkzeugen (XML-Edititor, MS-Word etc.). Durch den Roboter werden die dezentral erstellten Bestände in den zentralen Bestand übernommen.

Der Hauptnachteil der Architektur liegt darin, daß das System neuentwickelt werden muß, so daß es die Verwaltung strukturierter Dokumente im Mehrbenutzerbetrieb und Anfragen auf Basis der universellen Anfragesprache adäquat unterstützt. Das System muß große Datenbestände (Milliarden strukturierter Dokumente) performant verarbeiten können. Beim Einsatz des Roboters kann der Bestand des Systems nicht aktuell zur Verfügung gestellt werden. Eine Verzögerung um mehrere Tage ist möglich. Neue bzw. Änderungen an den Bestandsadressen müssen dem Roboter bekanntgegeben werden. Weiterhin muß die Konfiguration des Roboters geändert werden, falls sich die Schnittstelle zum externen System ändert.

Mit Standards in den Bereichen Dokumentbeschreibungssprachen (HTML, XML), Protokollen (HTTP, Corba-IIOP), Applikationsserver (J2EE) und Präsentationsserver können komplexe Client-Server Anwendungen der beschrieben Art performant und skalierbar realisiert werden. Zur Realisierung des Collection Manager and Query Server kann mittelfristig ein vorhandenes Informationssystem integriert bzw. weiterentwickelt werden (Oracle Intermedia, Hyper-G). Langfristig ist der Einsatz einer Datalog-Maschine wünschenswert.

6 Terminologiebasiertes Information Retrieval

Es ist bisher wenig erforscht, das thesaurusbasierte Information Retrieval zum terminologiebasierten Information Retrieval zu erweitern und in den Kontext der Hypertextforschung zu stellen. Die zugrundeliegende Repräsentationssprache muß dafür einerseits mächtig genug, andererseits auf die speziellen Belange vorhandener Terminologiebestände zugeschnitten sein.

6.1 Terminologien und Wörterbücher

Eine Terminologie im Kontext der Dokumentation und Terminologiewissenschaft ist als ein fachsprachliches Wörterbuch zu verstehen:
"Gesamtheit der innerhalb eines wissenschaftlichen Systems definierten Fachausdrücke, die sich von umgangssprachlicher Verwendung durch exakte Definition innerhalb eines bestimmten Systems unterscheiden." (Bußmann (1990)).

Ein **komplex strukturierter Thesaurus** ist ein *Thesaurus*, in dem mehrere Arten inhaltlicher *Begriffsbeziehungen* dargestellt werden.
DEF: 44-02, 11-01, 41-01
SYN: Thesaurus im engeren Sinne
E: complex structured thesaurus
F: thésaurus à structure complexe
ANM: 1. Dies ist die eigentliche Form des Thesaurus
2. Wird er im Hauptteil systematisch geordnet, spricht man von einem hierarchischen Thesaurus
BSP: TEST-Thesaurus
OB: Thesaurus
UB: Thesaurus ohne Vorzugsbenennungen
VB: Kontrollierte Schlagwortliste

Abbildung 43: Der Terminologieeintrag 44-21: nach DIN 2339

Terminologien sind Netze, die aus typisierten Einträgen und Beziehungen zwischen diesen bestehen.
Folgende Relationen können unterschieden werden (vgl. Felber, Budin (1989)):
Ontologische Relationen (Gegenstandsrelationen):
Bestandsrelation (räumlich), Nacheinander-Relation (zeitlich), Relation zum Stoffgebilde, Wirk-Relation, ...
Begriffliche Relationen:
Ober/Unterbegriffsrelation, Verwandtschaftsrelation, Antonymie, Logische Nebenordnung, Logische Diagonalbeziehung, Logische Begriffsleiterbeziehung, Logische Reihenbeziehung, ...
Relationen zur Begriffsbeschreibung und Erläuterung:
Inhaltsbeschreibung (Definition), Umfangsbeschreibung, Beispiel, Abbildung, Erläuterung, Kommentar, ...
Themenrelationen:
Ober/Unterthema, verwandtes Thema, Themaleiter, Themareihe, Überschneidung, ...
Relationen zwischen Begriffszeichen (Benennungen) und Begriffen:
Homonymie, Synonymie, Quasi-Synonymie, Wortform, Langform, Kurzform, ...
Übersetzungsrelationen:

Deutsch-Englisch, Deutsch-Japanisch, ...
Andere Relationen:
Produzent-Produkt, Sender-Empfänger, ...

Die terminologischen Relationen lassen sich anhand der Merkmale Hierarchie und Dimension in polyhierarchische und polydimensionale Relationen trennen.
Für polyhierarchische Relationen (wie beispielsweise die Ober/Unterbegriffsrelation) gilt, daß ein Knoten bezüglich dieser Relation sowohl mehr als einen direkt untergeordneten als auch mehr als einen direkt übergeordneten Knoten haben darf.
Für polydimensionale Relationen gilt, daß ein Knoten anhand von Unterteilungsgesichtspunkten (Merkmalarten) unterteilt wird.

Im Gegensatz zu Terminologien beziehen sich Wörterbücher nicht ausschließlich auf fachsprachliche sondern auf beliebige lexikalische Einheiten.
"*Das Wörterbuch ist eine durch ein bestimmtes Medium präsentierte Sammlung von lexikalischen Einheiten (vor allem Wörtern), zu denen für einen bestimmten Benutzer bestimmte Informationen gegeben werden, die so geordnet sein müssen, daß ein rascher Zugang zur Einzelinformation möglich ist.*" (Hausmann (1985)).
Wie Terminologien zeichnen sich Wörterbücher durch komplexe, wörterbuchspezifische Verweisstrukturen aus. Für Wörterbücher sind Typen lexikalischer Einheiten und Relationen zu unterscheiden, die hier nicht weiter aufgezählt werden.

6.2 Terminologiebasiertes Information Retrieval: Definition der Begriffe

Man kann das passive von dem aktiven terminologiebasierten Information Retrieval zu unterscheiden (Blair (1990) S. 56): Bei der passiven Methode werden zu dem Frageterm inhaltlich in Beziehung stehende Terme selektiert und dem Benutzer zur weiteren Fragestellung vorgestellt. Hier dient der Thesaurus dem Benutzer als Hilfe bei der Auswahl weiterer Suchbegriffe quasi als Erinnerungshilfe. Voraussetzung dabei ist, daß der Benutzer entscheiden kann, welche vorgeschlagenen Terme ein besseres Suchergebnis versprechen. Bei der aktiven Methode werden zu dem Frageterm inhaltlich in Beziehung stehende Terme automatisch der Anfrage hinzugefügt.
Welche inhaltlich in Beziehung stehende Terme herangezogen werden, kann festgelegt werden. So verspricht die Hinzunahme von hierarchisch übergeordneten Begriffen (Oberbegriffen, Besteht-Aus-Begriffen etc.) zu der ursprüngliche Anfrage ein breiteres Suchergebnis und die Beschränkung auf hierarchisch untergeordnete Begriffe (Unterbegriffe, Teilbegriffe etc.) in der Anfrage ein schmaleres Suchergebnis.
Homonyme und Synonyme können durch weitere Anfragen auf einen Begriff gebracht werden: Meinen Sie Nuß im Sinne von Obst, Pflanze, Kopf, als Teil des Gewehrschlosses oder als Geschlechtsteil eines Hundes?
Als Einstiegsvokabular für die Recherche sind für den Benutzer Basisbegriffe (vgl. Rosch (1978) S. 27-48: basic level concepts) besonders wichtig. Basisbegriffe werden in der natürlichen Sprache häufiger verwendet als abstrakte Begriffe oder sehr spezielle Begriffe. Beipiel: Meyer fragt Müller: "Wie geht es ihrem Hund?" Müller antwortet: "Noch nicht so gut, aber er frißt wieder." Meyer würde niemals fragen: "Wie geht es ihrem Lebewesen?" oder "Wie geht es ihrem Pinscher?"

Wichtig ist weiterhin, welchen Erfahrungshorizont ein Benutzer besitzt. Ein Experte eines Fachgebiets verwendet als Einstiegsvokabular seiner Recherche in der Unterbegriffshierarchie eher tieferliegende Basisbegriffe als beispielsweise ein Anfänger auf dem Gebiet.

Der Hauptvorteil des terminologiebasierten Retrievals besteht darin, daß der Informationssuchende während seiner Suche erkennt, wie die Informationsbestände durch die Terminologien strukturiert sind. Er bekommt terminologische Hilfestellungen angeboten, anhand derer er entscheidet, welches terminologische Vokabular bei seiner Suche Erfolg verspricht. Der Informationssuchende navigiert beispielsweise graphisch in dem Terminologienetz und wählt Einträge, die er in seiner Anfrage berücksichtigen will. Der Anfragende erkennt, welche Termini in einem Sachgebiet zur Suche vorhanden sind und welche inhaltlich verwandten Termini existieren.

Terminologiebasiertes Retrieval: Terminologiebasiertes Retrieval heißt der Prozeß, bei dem eine Anfrage mit terminologischen Einheiten eine Teilmenge des zugrundeliegenden Terminologie- und Informationsbestandes als Ergebnis zurückliefert. Eine Anfrage kann kombiniert als Kommando oder mit Hilfe von Navigations- und Darstellungswerkzeugen gestellt werden.

Für das terminologiebasierte Retrieval existieren folgende Aufgaben:

- Informationsbestände durch Terminologien strukturieren (indexieren). Fachwissen über die in einem Fachgebiet verwendeten Termini vermitteln. Definitionen und Beispiele geben (auch in multimedialer Form).
- Einstiegsgebiete in den Informationsbestand mit Hilfe der Terminologien aufzeigen.
- (Zwischen)-Ergebnisse präsentieren: Je ausführlicher die Suchergebnisse dargestellt werden, desto leichter ist es für den Nutzer, die nachgewiesene Information hinsichtlich ihrer Relevanz zu beurteilen, umso mehr Zeit kostet jedoch deren Sichtung.
- Darstellungs- und Navigationswerkzeuge entwickeln, die die Selektion, Analyse und Weitergabe terminologischer Information verbessern.

Zur Verbesserung des terminologiebasierten Retrievals können Morphologiekomponenten wie beispielsweise MOLEX vom IDS Mannheim, das in Saarbrücken entwickelte System MORPHIX oder Intermedia (Oracle (2000a) und Oracle (2000b)) eingesetzt werden. Mittels einer solchen Komponente wird eine Wortform auf ihre Grundform zurückgeführt bzw. aus einer Grundform alle Wortformen bestimmt.

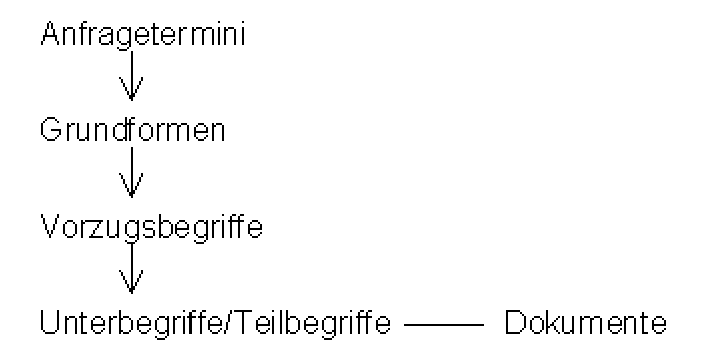


Abbildung 44: Suche nach Dokumenten mit Morphologie- und Terminologiekomponente

Wenn ein Volltextzugriff auf Dokumente erreicht werden soll können nach Bestimmung der Unterbegriffe bzw. Teilbegriffe noch Synonyme und Vollformen mit in die Suche einbezogen werden, um das Suchergebnis zu verbessern

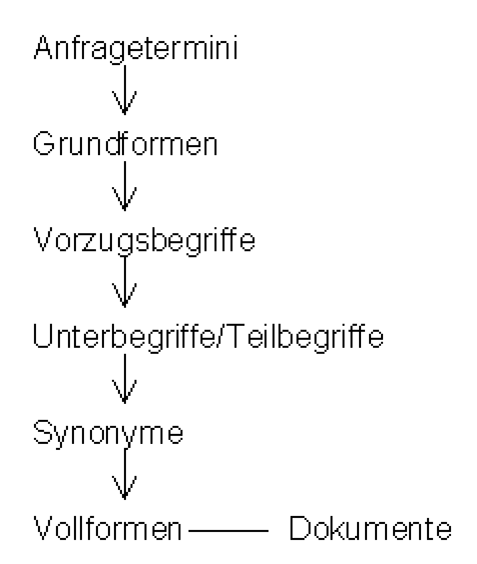


Abbildung 45: Volltextsuche nach Dokumenten mit Morphologie- und Terminologiekomponente

Das Wechselspiel kommandoorientierter Anfragenformulierung und der menügeführten Suche in Dokumentenbeständen stellt einen wichtigen Forschungsgegenstand dar. Beispielsweise kann die Strategie gewählt werden, mit einem Anfragekommando einen Einstiegspunkt in den Dokumentenbestand zu gewinnen, um dann mit einer menügeführten Suche die Informationsmenge weiter einzuschränken. Solche Einstiegsfragen sind bespielsweise: "Welche Sachgebiete existieren auf dem Gebiet des Information Retrieval?" oder "Welche globalen Informationsbestände existieren für das Gebiet des Information Retrieval?"
Die umgekehrte Strategie besteht darin, mit Hilfe der menügeführten Suche ein Gefühl für die Begrifflichkeiten des Fachgebietes zu entwickeln, um dann mit gezielten Anfragekommandos die gewünschte Information abzurufen.

Dem Informationssuchenden sollte bei seiner Anfragenvor- und -nachbereitung Unterstützung angeboten werden. Anfrageergebnisse werden für die zukünftige Suche in den Informationsbeständen festgehalten. Weiterhin können Relevanzfeedbackverfahren eingesetzt werden (Salton, McGill (1987) S. 255).
Weiterhin können häufig verwendete Suchtermini erfaßt und dem Informationssuchenden als Kandidaten zur Verfügung gestellt werden. Zusätzlich können die Verwendungshäufigkeiten der Anfragetermini in dem Information Retrieval System als externe Kandidaten berücksichtigt werden (Anpassung des Thesaurus an die Benutzer).

6.3 Konsistenz und Inferenz

Formale Eigenschaften von Relationen lassen sich zur Konsistenzsicherung von Terminologie- und Dokumentbeständen nutzen. Die Konsistenz der Bestände wird anhand der formalen Eigenschaften und der Argumenttypen (gegeben durch das kartesische Produkt) der Relationen überprüft. Sie wird dadurch unabhängig von dem Namen und der intendierten aber nicht festgelegten Semantik der Relationen. Ähnliche Ansätze finden sich in Lenat (1990) und Schönfeldt (1994).

Konsistenz durch formale Eigenschaften von Relationen

$R \subseteq M_1 \times M_2$; $x_1,...,x_n$, x, y, z $\in M_1, M_2$

R reflexiv $:\Leftrightarrow \forall x\ (x\ R\ x)$

R irreflexiv $:\Leftrightarrow \forall x\ \neg(x\ R\ x)$

R symmetrisch $:\Leftrightarrow \forall x\ \forall y\ ((x\ R\ y) \Rightarrow (y\ R\ x))$

R asymmetrisch $:\Leftrightarrow \forall x\ \forall y\ ((x\ R\ y) \Rightarrow \neg(y\ R\ x))$

R antisymmetrisch $:\Leftrightarrow \forall x\ \forall y\ ((x\ R\ y) \wedge (y\ R\ x) \Rightarrow (x = y))$

R transitiv $:\Leftrightarrow \forall x\ \forall y\ \forall z\ ((x\ R\ y) \wedge (y\ R\ z) \Rightarrow (x\ R\ z))$

R atransitiv $:\Leftrightarrow \forall x_1 \ldots \forall x_n\ ((x_1\ R\ x_2) \wedge \ldots \wedge (x_{n-1}\ R\ x_n) \Rightarrow \neg(x_1\ R\ x_n))$ für $n>2$

R äquivalent $:\Leftrightarrow$ ((R reflexiv) $\wedge$ (R symmetrisch) $\wedge$ (R transitiv))

R linkseindeutig $:\Leftrightarrow \forall x\ \forall y\ \forall z\ ((x\ R\ y) \wedge (z\ R\ y) \Rightarrow (x = z))$

R rechtseindeutig $:\Leftrightarrow \forall x\ \forall y\ \forall z\ ((x\ R\ y) \wedge (x\ R\ z) \Rightarrow (y = z))$

R eineindeutig $:\Leftrightarrow$ ((R linkseindeutig) $\wedge$ (R rechtseindeutig))

R linkstotal $:\Leftrightarrow \forall x\ \exists y\ (x\ R\ y)$

R rechtstotal $:\Leftrightarrow \forall y\ \exists x\ (x\ R\ y)$

R bitotal $:\Leftrightarrow$ ((R linkstotal) $\wedge$ (R rechtstotal))

R funktion $:\Leftrightarrow$ ((R linkstotal) $\wedge$ (R rechtseindeutig))

R injektiv $:\Leftrightarrow$ ((R linkstotal) $\wedge$ (R eineindeutig))

R surjektiv $:\Leftrightarrow$ ((R bitotal) $\wedge$ (R rechtseindeutig))

R bijektiv $:\Leftrightarrow$ ((R bitotal) $\wedge$ (R eineindeutig))

R zyklisch $:\Leftrightarrow \forall x_1 \ldots \forall x_n\ ((x_1\ R\ x_2) \wedge \ldots \wedge (x_{n-1}\ R\ x_n) \wedge (x_n\ R\ x_1))$

R azyklisch $:\Leftrightarrow \forall x_1 \ldots \forall x_n\ \neg((x_1\ R\ x_2) \wedge \ldots \wedge (x_{n-1}\ R\ x_n) \wedge (x_n\ R\ x_1))$

Konsistenz durch Argumenttypen

Für die Argumente jeder einzelnen Beziehung wird überprüft, ob sie vom Typ her den Argumenten (gegeben durch das kartesische Produkt der Typen) der beteiligten Relation entsprechen.

Zusammen mit Schönfeldt (1994) wird die Anwendung formaler Eigenschaften auf zweistellige terminologische Relationen untersucht. Diese Untersuchung wird hier zusammengefaßt:

Relation	refl.	irrefl.	sym.	antisym.	trans.	atrans.
Strenge-Synonymie	ja	nein	ja	nein	ja	nein
Quasi-Synonymie	ja	nein	ja	nein	nein	nein
Abstraktionsrelation	nein	ja	nein	ja	ja	nein
Bestandsrelation	nein	---	nein	ja	---	nein
Functional component	nein	nein	nein	ja	nein	nein
Segmented whole	nein	ja	nein	ja	ja	nein
Collection member	nein	ja	nein	ja	nein	nein
Set-subset	nein	nein	nein	ja	ja	nein

Logische Gleichordnung (monohierarchisch)	nein	nein	ja	nein	ja	nein
Logische Gleichordnung (polyhierarchisch)	nein	nein	ja	nein	nein	nein
Räumliche Nebenordnung	nein	nein	ja	nein	nein	nein
Kausalbeziehung (direkt)	nein	ja	nein	ja	nein	ja
Kausalbeziehung (indirekt)	nein	ja	nein	ja	ja	nein
Nachfolgebeziehung (direkt)	nein	nein	nein	nein	nein	ja
Nachfolgebeziehung (indirekt)	nein	nein	nein	nein	ja	nein
Antonymie	nein	ja	ja	nein	nein	ja

Tabelle 11: Formale Eigenschaften terminologischer Relationen

Außer den hier betrachteten Eigenschaften lassen sich den zweistelligen terminologischen Relationen weitere bei Schmidt, Ströhlein (1993) S. 29ff genannte formale Eigenschaften zuordnen.
Für ein Terminologie-Navigationssystem könnten folgende Regeln für Relationen festgelegt werden:
1. Jede zweistellige Relation besitzt eine converse Relation (Vermeidung von Fallgruben).
2. Jede symmetrische Relation besitzt als converse Relation sich selbst.
3. Es existieren bezüglich einer Relation keine Mehrfachbeziehungen zwischen zwei Knoten.
4. Für jede hierarchische, polyhierarchische oder polydimensionale Relation gilt die Azyklizität und die Atransitivität.

Inferenz
Mit Hilfe logisch definierter Regeln lassen sich implizite Informationen aus den Terminologien und den Informationsbeständen gewinnen (vgl. auch terminologische Logiken).
"Im Unterschied zu gewöhnlichen Datenbanksystemen, wo im allgemeinen nur explizit gespeicherte Informationen abgerufen werden können, liefert ein Frage-Antwort-System (Bem.: oder wissensbasiertes System) auch implizite Informationen, d.h. Informationen, die sich logisch deduzieren lassen." (Konrad (1986))

Beispiele für solche Regeln sind:
- Generierung von Beziehungen:
  $\text{Conn}(x_1, x_3, r)$ gdw $\text{Conn}(x_1, x_2, r) \wedge \text{Conn}(x_2, x_3, r) \wedge \text{Prop}(r, \text{"transitiv"})$
- Vererbung von Attributen entlang einer spezifizierten Relation (z.B. bei transitiven Relationen)
- Generierung von conversen Relationen

- Definition von Relationen anhand von Relationen: z.B. großvater = vater(vater) $\vee$ vater(mutter)

weitere siehe Lenat (1990).

6.4 Entwicklung eines Kommunikationsmodells für das terminologiebasierte Information Retrieval

Traditionell wird die Ablage von Information als Dialog zwischen Informationsablegenden (Indexierer, Terminologe, Terminologiepfleger, Autor, Indexiersystem etc.) und die Suche nach Information als Dialog zwischen dem Informationssuchenden (Suchlaie, Informationsvermittler etc.) und dem Anbieter von Information (Verkäufer, Bibliothekar, Informationssystem etc.) geführt. Nachrichten (Fragen und Antworten) werden in der gemeinsamen Dokumentationssprache ausgedrückt Unabhängig von der kommunikativen Situation der Kommunikationspartner wird eine Frage gestellt und eine Antwort gegeben. Gegebenenfalls wird eine Anfrage durch Relevanzfeedbackverfahren verfeinert.

Terminologiebasiertes Information Retrieval soll hier als eine komplexe Kommunikationshandlung verschiedener Arbeitsgruppen verstanden werden, in der in wechselseitigem Dialog Information abgelegt, nach Information gefragt und Information als Antwort gegeben wird. Damit das nicht im Chaos endet, kommunizieren die Arbeitsgruppen über ein ortsunabhängiges, gemeinsames Kommunikationsbrett (schwarzes Brett) für das zu bearbeitende Fachgebiet. Weiterhin wird eine Versionskontrolle (Kandidaten, Ein/Aus-Check) eingeführt und eine Rollenverteilung für das redaktionelle Arbeiten im Team vorgenommen.

Indexierer, Autor:

Der Indexierer versucht Dokumente formal und inhaltlich so zu katalogisieren, daß der Informationsuchende (und Terminologe) sie wiederfindet. Er ist verantwortlich für die Indexierung der Informationsbestände und greift kommunikativ in den Prozeß der Terminologiepflege und Informationssuche ein.

Er schlägt möglichst zusammen mit dem Autor des neuen Dokuments einen Katalogeintrag am schwarzen Brett vor. Dieser wird dort gegebenenfalls von anderen Indexierern verbessert und von Informationssuchenden und Terminologen/Terminologiepflegern kommentiert und in einem speziellen Modus des Informationsystems getestet.

Nach Ablauf einer solchen Diskussions- und Testphase legt der Indexierer den von ihm gegebenfalls modifizierten Katalogeintrag in dem Katalog ab.

Der Indexierer schlägt weiterhin Kandidaten für zu modifizierende oder zu entfernende Katalogeinträge am schwarzen Brett vor, die von ihm wiederum nach einer Diskussions- und Testphase in dem Katalog abgelegt werden.

Eine weitere Aufgabe des Indexierers besteht darin, am schwarzen Brett Kandidaten für neue, zu modifiziernde oder zu entfernende Terminologieeinträge vorzuschlagen. Diese werden von Terminologen/Terminologiepflegern verbessert und von Informationssuchenden und anderen Indexierern am Fachgebiet kommentiert und getestet.

Weiterhin untersucht der Indexierer von den anderen Indexierern und Arbeitsgruppen vorgeschlagene Terminologie-Kandidaten in der Praxis und beurteilt sie am schwarzen Brett.

Der Indexierer beurteilt zudem die Relevanzurteile, die Informationssuchende für ihre Anfragen am schwarzen Brett veröffentlicht haben und stellt daraus resultiernde Modifikationswünsche für den Terminologiebestand und den Katalog am schwarzen Brett vor.

Der Indexierer legt sein Wissen über Terminologiepflege und Indexierung offen (Begründungen, nach denen er in der Praxis vorgeht).

Informationssuchender, Informationsvermittler:
Informationssuchende versuchen ihre Anfragen so zu formulieren, daß dies der Indexierer (und Terminologe) versteht. Der Informationssuchende ist verantwortlich für die Formulierung seiner Anfrage.
Wenn der Informationssuchende an einer Verbesserung seiner Kommunikation mit den Informationsablegenden interessiert ist, kann er über das schwarze Brett mit ihnen in Kontakt treten und damit seinen Einfluß bei der Informationsablage geltend machen:
Er testet das zugrundeliegende System dadurch, daß er dort öffentlich die Relevanz der Dokumente, die als Antwort seiner Anfrage gegeben wurden, beurteilt. Kandidatenvorschläge werden besonders hervorgehoben. Dem Informationsablegenden wird durch die Analyse des Testergebnisses die Möglichkeit gegeben, Fehler in der Indexierung und Terminologieerstellung zu beheben.
Weiterhin kann der Informationssuchende am schwarzen Brett allgemeine Probleme bei der Recherche und allgemeine Verbeserungsvorschläge diskutieren und allgemeine Fragen stellen.

Er kann Kandidatenvorschläge der Informationsablegenden (Terminologieeinträge, Katalogeinträge) kommentieren.
Der Informationssuchende begründet seine Suchstrategie bei der Recherche.

Terminologiepfleger, Terminologe:
Terminologiepfleger versuchen Terminologien so zu erstellen und zu verwalten, daß Informationssuchende und Indexierer diese verstehen und für das Information Retrieval effizient einsetzen können.
Der Terminologiepfleger schlägt Kandidaten für neue, zu modifizierende und zu entferndende Terminologieeinträge am schwarzen Brett vor. Nach Ablauf einer Diskussions- und Testphase legt er den Kandidaten in den Terminologiebestand ab. Gegebenenfalls überprüft er zusammen mit den Indexierern die Konsistenz des Terminologie- und Informationsbestandes.
Ein weitere Aufgabe des Terminologiepflegers besteht darin, Verbesserungen von weiteren Kandidaten vorzuschlagen oder Kandidaten zu kommentieren.
Der Terminologiepfleger begründet seine Strategie der Terminologiepflege.

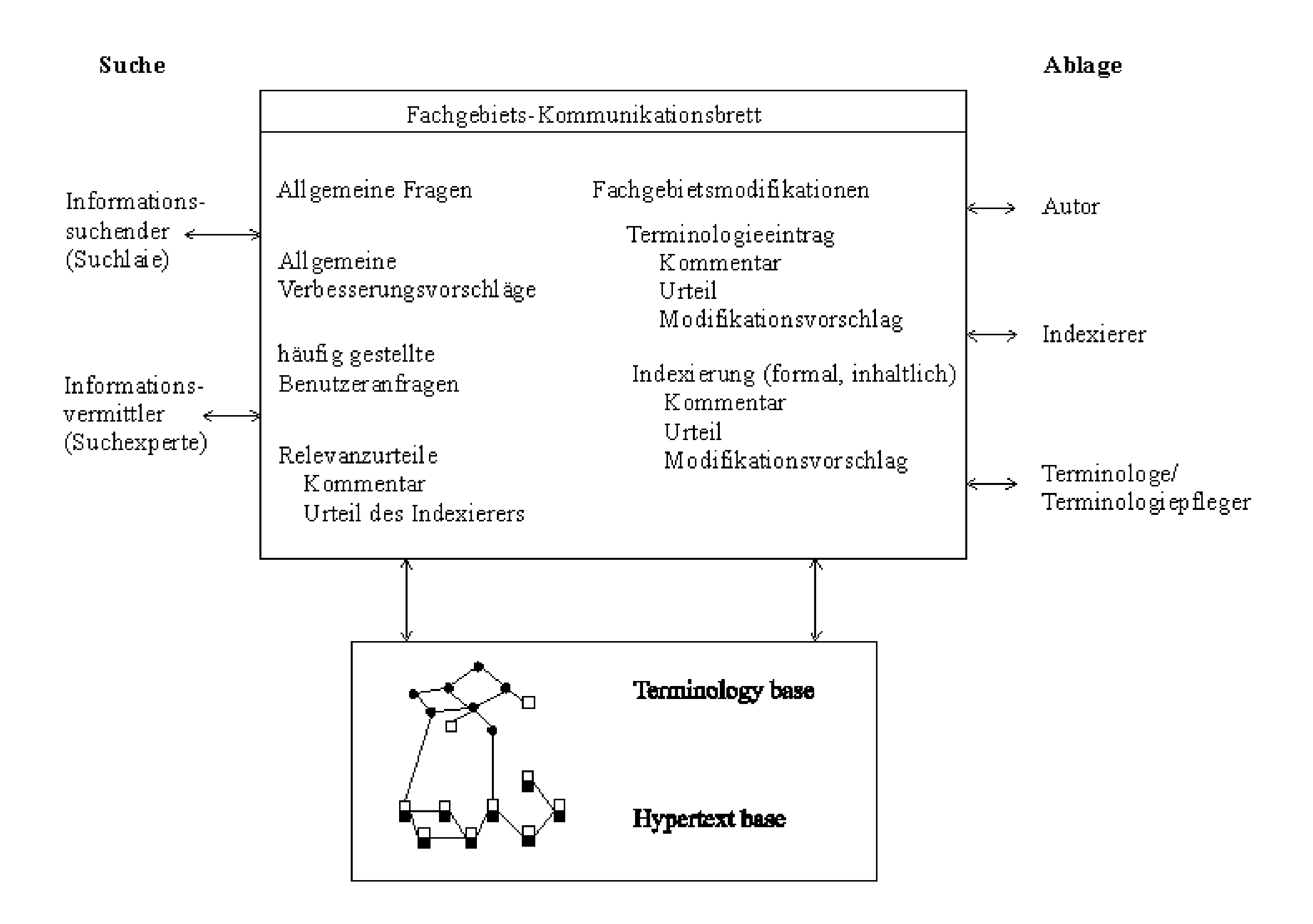


Abbildung 46: Kommunikation bei der Informationssuche und -ablage

7 Schluß

7.1 Vergleich von Anfragen in Internet-Informationssystemen

Auf Phasen der Diversifikation von Systemen folgen Phasen ihrer Integration und Vereinheitlichung. Diese Regel trifft zur Zeit auf Internet-Informationssysteme zu. Benutzer müssen hintereineinander verschiedene Systeme anwenden, um ein bestimmtes Dokument im Internet zu finden. Sie müssen Spezialwissen über diese Systeme erlangen: verwendete Anfragesyntax und -semantik, Dokumentbestand, Suchoberfläche, Anzeige des Suchergebnisses etc. Metasuchmaschinen versuchen zwar dieses Problem zu verringern, stellen jedoch keine grundsätzliche Lösung dar.

Bei dem Vergleich der Internet-Informationssysteme fällt auf, daß sich die Systeme insbesondere unterschiedlicher Systemtypen auf Dokumenttypen spezialisieren. Beispiele sind Suchmaschinen, die sich auf HTML- oder News-Dokumente beschränken. Gründe dafür liegen in der fehlenden Normierung beispielsweise in Form von XML-DTD's für die Dokumenttypen und in der schwierigen und teuer zu realisierenden Spezialbehandlung. Der Vorteil der Spezialisierung liegt jedoch darin, daß die Suche spezialisiert durchgeführt werden kann und Eigenheiten der Dokumenttypen besonders behandelt werden können. Parallele Entwicklungen fördern zudem Innovationen heraus. Nachteile der Spezialisierung sind darin zu sehen, daß die Retrievaleffektivität (Recall-Ratio, Precision) niedriger als in einem universellen System ist. Die Recall-Ratio ist geringer, da der Dokumentbestand kleiner ist. Die Precision ist höher, da der Dokumentbestand kleiner ist und die Wahrscheinlichkeit, daß sich Dubletten im Bestand befinden, geringer ist. Zur Verbesserung der Retrievaleffektivität in universellen Informationssystemen können verschiedene Mittel eingesetzt werden: Verwendung eines Thesaurus, Verbesserung der Ähnlichkeitsfunktionen z.B. für geschachtelte Dokumente, Einsatz von Relevanzfeedbackverfahren, logische Zusammenfassung komplexer Suchergebnisse etc. Diese müssen jedoch in der Praxis getestet werden. Die Bestimmung der Retrievaleffektivität in großen, dynamischen Informationssystemen ist jedoch generell problematisch (vgl. Salton (1975) S. 236-237).

Das Problem der Überlappung des Bestandes von Internet-Informationssystemen liegt in der Architektur der Systeme. Eine Suchmaschine wie Google mit einem Bestand von ca. 1,2 Mrd. Dokumenten verwendet einen anderen Gathering-Mechanismus als z.B. Fast mit einem Bestand von 0,57 Mrd. Dokumenten. Weiterhin spielen bei der Auswahl des Bestandes kommerzielle Interessen eine große Rolle.

Oft unterscheiden sich Internet-Informationssysteme syntaktisch stark (Anfragesprache, Benutzeroberfläche) aber semantisch wenig voneinander. Dies zeigt sich z.B. innerhalb der Klasse der Suchmaschinen recht deutlich. Beispielsweise wird dort "near(5)" und "near 5" bzw. "and" und " " jeweils für dieselbe Anfrage verwendet. Weiterhin werden stark unterschiedliche Benutzeroberflächen für dieselben Anfragen bereitgestellt. In manchen Systemen z.B. eine "einfache Suche" in ein anderes Fenster als die "Expertensuche" gestellt. In anderen Systemen werden diese beiden Sucharten in ein Fenster zusammengelegt.

Die unterschiedliche Definition der Semantik ist beispielsweise bei den Ähnlichkeitsfunktionen zu beobachten. Bei Goggle ergibt die Anfrage "Hund Katze" ein anderes Ergebnis als bei Alta-Vista.

Bei der Integration von Internet-Informationssystemen ergeben sich Schwierigkeiten folgender Art:
1. Erfassung, Bestand:
- Global einheitliche formale und inhaltliche Beschreibung der Dokumente
- Internationaliserung: Zeichensätze, Terminologien, Abbildung der Benutzeranfragen etc.
- Starkes Anwachsen des Dokumentenbestands
- Beseitigung der überlappenden Bestände (Dublettenbeseitigung)
- Großer Anteil an Dokumenten, die die Veröffentlichungshürde nicht überwinden würden: Notizzettel, Test-Dokumente etc.

2. Suchmittel:
- Realisierung effizienter regulärer Volltextanfragen
- Unterschiediche Definition der Ähnlichkeitsfunktionen
- Integration unterschiedlicher Systemtypen (insbesondere Suche in link-strukturierten Dokumenten, Suche in geschachtelten Dokumenten, Faktensuche und intelligente Agenten)
- Personalisierung

3. Suchergebnisse
- Integration unterschiedlicher Systemtypen
- Geordnete Menge von Dokumenten versus Netz von Dokumenten

Es fällt auf, daß sich einige Systeme mehr in Richtung einer größeren Universalität der Anfragemöglichkeiten entwickeln. Intermedia vereinheitlicht den Zugriff auf einfache und geschachtelte Dokumente, insbesondere auf Basis von XML und anderer Dokumenttypen. Es ist das wohl mächtigste System dieser beiden Systemklassen, was die Ausdrucksfähigkeit und Semantik angeht. Gleichzeitig ist es technisch ausgereift und für größere Dokumentbestände ausgelegt. Dagegen ist die Integration der Suche in link-strukturierten Dokumente noch nicht sehr weit fortgeschritten. Erste Systeme existieren zwar, doch sind diese ausschließlich auf die Suche in link-strukturierten Dokumenten spezialisiert und sind technisch noch nicht für größere Bestände ausgelegt. Das System Hyper-G/Hyperwave verspricht mittelfristig am ehesten eine Lösung des Problems.
Ein wichtiges Forschungsgebiet für die Erweiterung und Integration von Internet-Suchmöglichkeiten ist die Frageforschung (natürliche und künstliche Anfragen[8]). Als Ergebnis ist eine Klassifikation von Informationssystemen nach Anfragemächtigkeit zu erwarten. Dadurch würde die Entwicklung übergreifender Suchmöglichkeiten beschleunigt und Doppelarbeit bei der Entwicklung von Informationssystemen vermieden.

7.2 Entwickelte Anfragesprachen

Es wurden Anfragesprachen für drei Systemklassen (Suche in einfach strukturierten Dokumenten, Suche in link-strukturierten Dokumenten und Suche in geschachtelten Dokumenten) entwickelt und zu einer universellen Anfragesprache zusammengeführt. Die meisten Beispielanfragen aus Kap. 3 können mit den Anfragesprachen abgebildet werden. Darüberhinaus sind mit StructuredQL universelle Anfragen zur Suche in strukturierten Dokumenten möglich.

[8] siehe auch Umstätter, Walter. Die Kunst der Frage. In: Bibliotheksdienst 27, S. 1180-1191, 1993.

Im Vergleich der universellen Anfragesprache StructuredQL mit anderen logischen Anfragesprache fällt auf, daß im Beeri-Kornatzky-Modell Quantoren (modale Quantoren) und Pfade ausgedrückt werden können.
IQL bietet die Suche mit Attributen, Volltext- und Thesaurusoperatoren und drückt Fakten und Erklärungen aus. Operatoren für die Suche in link-strukturierten Dokumenten und die Suche in geschachtelten Dokumenten werden nicht abgebildet.
Das Datalog-Modell von Fuhr modeliert Netzwerkoperatoren und Operatoren für Teile mit der Sprache Datalog. Datalog wird seit langem in der Forschung diskutiert und es existieren mehrere Implementierungen der Sprache. Das Fuhr-Modell kann relativ einfach durch Programmierung der gewünschten Sprachelemente erweitert werden.
StructuredQL modelliert die Suche in einfach strukturierten Dokumenten (Suche mit Attributen, Sorten von Individuenbereichen), die Suche in link-strukturierten Dokumenten (Netzwerkoperatoren) und die Suche in geschachtelten Dokumenten (Operatoren für Teile/Behälter). Zusätzlich werden komplexe boolesche Operatoren bereitgestellt. Weiterhin werden Operatoren zur Volltextsuche, linguistische Operatoren und Thesaurusoperatoren abgebildet.
Der größte Vorteil bei der Entwicklung einer universellen Anfragesprache für Internet-Suchsysteme liegt darin, einheitlich auf Internet-Bestände zugreifen zu können. Gleichzeitig bleibt die bisherige Anfragemächtigkeit erhalten. Die Retrievaleffektivität im Internet kann insgesamt verbessert werden. Die Benutzung universeller Anfragemöglichkeiten ist in der Regel einfacher als parallel oder hintereinander auf mehrere unterschiedliche Systeme mit ihren jeweiligen Anfragemöglichkeiten inklusive Benutzeroberfläche, Syntax und Semantik der Anfragesprachen, Dokumenttypen, Dokumentbestand und Suchergebnissen zuzugreifen. Der Benutzer hat mehr Zeit für seine eigentliche Arbeit und steigert letztendlich seine Zufriedenheit. Weiterhin werden durch den Einsatz einer universellen Anfragesprache Redundanzen und Überlappungen zwischen den Beständen vermieden. Falls die Anfragemächtigkeit insgesamt erhöht wird, mußte sie bisher parallel in mehreren Systemen nachgezogen werden. Dieser Mehraufwand entfällt bei dem Einsatz einer universellen Anfragesprache. Bei der Vereinheitlichung der Anfragemöglichkeiten darf das Innovationstempo jedoch nicht die Entwicklung neuer Suchsysteme durch zuviel Standardisierungsdruck aufhalten. Neuentwicklungen sollen und müssen weiterhin Anfragemöglichkeiten zuerst in Teilbereichen verbessern und sind erst danach in bestehende globale Informationssysteme zu integrieren.
Nachteil bei der Entwicklung einer übergreifenden, einheitlichen Lösung ist die Komplexität der Aufgabe. Bisherige Anfragemöglichkeiten dürfen nicht verlorengehen, Nutzer dürfen nicht überfordert werden und bei der Auftrennung in Nutzerklassen muß auf spezifische Nutzerbedürfnisse Rücksicht genommen werden. Weiterhin müssen vorhandene, schon erfaßte Dokumentbestände konsistent in das System integriert werden. Die Anfragesprachen müssen verfeinert werden und den Praxistest bestehen.

7.3 Architektur eines universellen Internet-Informationssystems

Die zentral/dezentrale Architektur eines universellen Internet-Informationssystems hat die größten Vorteile. Sie bietet insbesondere die volle Abbildung der gewünschten Anfragemächtigkeit durch den Einsatz einer universellen Anfragesprache zur Suche in strukturierten Dokumenten. Wichtigste Komponenten der Architektur sind Collection Manager, QueryServer und Roboter. Im Vergleich zu Standardsuchmaschinen wird zusätzlich eine direkte Verwaltung von Dokumenten im zentralen System durch den Benutzer ermöglicht. Weiterhin kann der Benutzer den Dokumentenbestand, in dem er recherchieren möchte, auswählen (lokal bis global). Spezifische Suchergebniseinstellungen sind möglich.

Zur Realisierung des Systems können Techniken zum Einsatz kommen, die das System skalierbar halten und so die Performance des Gesamtsystems an die Benutzerzahl anpassen. Im Softwarebereich wird dies beispielsweise durch Komponententechnik (z.B. Enterprise Java Beans etc.), durch Programmierbibliotheken zur verteilten Client-Server-Entwicklung und durch den Einsatz von Applikationsservern erreicht. Im Hardwarebereich wird Skalierbarkeit durch die Verteilung der Hardwareressourcen erreicht. Der Datenbestand kann in Zukunft durch den Einsatz leistungsfähiger Hintergrundspeicher auf eine Größe im Petabyte-Bereich und das Cache-System durch Clusterlösungen auf eine Größe im Terabyte-Bereich anwachsen. Durch den Einsatz von Hochleistungssystemen (parallele Prozessoren, performante Datenbusse etc.) kann zudem die Verarbeitungsgeschwindigkeit erhöht werden, so daß Hunderttausende von Benutzern gleichzeitig mit dem System arbeiten können. Durch den Einsatz sogenannter Failover-Systeme werden Ausfälle vermieden. Datenbestände werden gespiegelt, beim Ausfall einzelner Hardwarekomponenten übernehmen andere Hardwarekomponenten automatisch, Hardware wird im laufendem Betrieb erweitert und ausgetauscht.
Das System wird mit offenen Schnittstellen realisiert, so daß Fremdsysteme eingebunden und Erweiterungen einfach durchgeführt werden können. Als Protokolle kommen Standard-Internet-Protokolle (HTTP, IIOP, RMI, FTP, NEWS, Z39.50, JDBC etc.) und als Austauschformate Standard-Internet-Austauschformate (XML, HTML, MS-Word-Doc etc.) zum Einsatz.
Wünschenswert ist eine Erweiterung des Systems zur Unterstützung der Faktensuche, der Suche nach terminologischen Einträgen und weiterer Sucharten im Bereich intelligenter Agenten (z.B. Relevanzfeedbackverfahren, personalisierte Suche etc.). Dafür muß die universelle Anfragesprache für diese Belange erweitert werden und Änderungen im System nachgezogen werden.
Durch den Einsatz von Techniken aus Volltextsuchsystemen kann die Volltextsuche performant im Internet-Massenzugriff bereitgestellt werden (siehe Suchmaschinen etc.). Die performante Realisierung der Suche in strukturierten Dokumenten mit hunderttausenden gleichzeitigen Nutzern muß noch geleistet werden und bleibt mittelfristig eine spannende Aufgabe. Zwischenlösungen für kleine bis mittlere Nutzerzahlen sind jedoch schon jetzt in skalierbarer Architektur machbar.

7.4 Terminologiebasiertes Information Retrieval

Der Zugriff auf Information mit Hilfe von Terminologien bereitet Schwierigkeiten, weil die in den Fachgebieten erstellten Terminologie- und die damit verbundenen Dokumentenbestände in unterschiedlichen Formaten vorliegen, deren Interpretationen sich voneinander unterscheiden. Qualitativ hochwertiges und effektives terminologiebasiertes Information Retrieval erfordert die Entwicklung einheitlicher und übergreifender Anfrageverfahren, die die Vielfältigkeit und Komplexität der Bestände berücksichtigen.
Die Verwendung von Thesauri und anderer Wissensbasen beim Information Retrieval erlebt in letzter Zeit einen Aufschwung vor allem im Internetbereich, da sie eine Verbesserung der Retrievalergebnisse verspricht. Dies kann nur dann erreicht werden, wenn die Wissensbasen qualitativ hochwertig aufgebaut und konsistent gepflegt werden. Mit der vorgestellten Methode, Relationen formale Eigenschaften zuzuordnen, wird die Konsistenz unabhängiger von dem Namen der Relation und ihrer implizit intendierten Semantik. Wenn man beispielsweise der Unterbegriffsrelation die Eigenschaften irreflexiv, antisymmetrisch und transitiv zuordnet, können keine selbstbezüglichen Verweise und keine redundanten Beziehungen entstehen. Die Qualität des Bestands verbessert sich. Voraussetzung ist allerdings, daß das Werkzeug zur Verwaltung der Wissensbasis diese Mechanismen adäquat und benutzerfreundlich unterstützt. Anhand eines objektorientierten Prototypen für die Thesauruspflege Thesauruspflege (vgl.

Willenborg (1993, 1994a) wurde die Machbarkeit gezeigt. Es wurde ein Editor zur Verwaltung mathematischer Eigenschaften von Thesaurusrelationen entwickelt und zur konsistenten Thesauruspflege bereits eingesetzt. Inkonsistenzen in vorhandenen Thesauri wie beispielsweise im Umweltthesaurus des Umweltbundesamtes konnten aufgedeckt werden.
Durch die Weiterentwicklung dieses Ansatzes konnten weitere Ergebnisse erzielt werden (vgl. Kap. 6.3). Es konnten allgemeinere Eigenschaften von Relationen (Eigenschaften von Eigenschaften von Relationen) aufgestellt werden. Weiterhin konnten Inferenzregeln entwickelt werden, die aus der Wissensbasis Wissen ableiten.
Die Nachteile liegen vor allem in dem relativ hohen Anfangsaufwand begründet. Es muß ein adäquates Pflegewerkzeug entwickelt werden. Zusätzliche Arbeit wird auch bei der Pflege der Wissensbasis und bei der Schulung der Mitarbeiter erzeugt. Dieser Aufwand erscheint jedoch angesichts der Verbesserung der Retrievalqualität vertretbar.
Die Entwicklung eines Modells, das die Kommunikation zwischen Informationssuchenden, Indexierern und Terminologiepflegern unterstützt und ihre Umsetzung in Form eines Computerwerkzeugs läßt folgende Verbesserungen erwarten:
a) Angleichung des Wortschatzes von Informationssuchenden, Indexierern und Terminologiepflegern. Dadurch wird der Verständigungsprozeß verbessert und letztendlich eine Verbesserung der Retrievalqualität erzielt.
b) Abbau der Trägheit bei der Terminologiepflege und Indexierung. Neuerungen können rascher diskutiert und in die Terminologie aufgenommen werden.
c) Erhöhung der Konsistenz bei der Indexierung. Veraltete Begrifflichkeiten werden zusammen mit den Terminologiepflegern rascher durch adäquate Begriffe, die der Informationssuchende versteht, ausgetauscht.
Der alleinige Kommandozugriff auf Information reicht i.A. nicht aus. Das Wechselspiel von Darstellung und Navigation trägt zur Verbesserung der Retrievaleffizienz bei. Der Benutzer blättert im Wissensbestand und gewöhnt sich an die verwendeten Begriffe und Bezeichnungen. Er tritt gleichsam in einen Dialog mit den Erstellern der Wissensbasis. Spielerisch erlangt er ein Gefühl für den Bestand. Die Größe des Bestands, interessante Fachgebiete etc. werden erkundet und in das Gesamtbild der Wissensbasis eingeordnet. Dabei können sich jedoch auch Probleme (Orientierungsverlust etc.) ergeben, die den erwarteten Mehrwert ins Gegenteil kehren.
Bei dem Aufbau der Wissensbasen ist ein Mittelweg zwischen Neuerung und Bewahrung anzustreben, so daß das vom Benutzer aufgebaute Verständnis nicht zu häufig durcheinander gerät. Darstellungs- und Navigationsmethoden dürfen aus diesem Grunde nicht zu häufig umgestellt werden.
Insgesamt werden weitere Ziele für das terminologiebasierte Information Retrieval erkennbar:
1. Vorhandene Bestände nutzen:
Es gilt, den großen Schatz vorhandenen Wissens ohne aufwendige Zwischenarbeiten in die eigene Wissensbasis zu übernehmen. Thesaurus Guide (1985) listet ca. 1000 Thesauri in Europa auf.
"*Da die Erstellung von Wissensbasen teuer ist, sollte größeres Augenmerk auf die Wiederverwendbarkeit von Wissensbasen gelegt werden. (...) Die Entwicklung großer Expertensysteme wird mit durchschnittlich einer Million Dollar veranschlagt. (...) Für zahlreiche Anwendungen wären Wissensbasen sehr nützlich, wenn sie billig und in einfacher Form zur Verfügung stünden.*"(Czedik (1992))
Für das Zusammenführen von Thesauri sind Konsistenzregeln einzuhalten.

2. Vorhandene Bestände anreichern:

Multilingual: Einsprachige Terminologiebestände werden mit speziellen Übersetzungsrelationen angereichert in die gewünschten Sprachen übersetzt. Dieser Mehraufwand amortisiert sich dadurch, daß Fremdsprachler auf die Informationsbestände zugreifen können.
Multimedial: Traditionell textuelle Terminologiebstände werden an den Stellen durch multimediale Elemente ergänzt, an denen dies einen Mehrwert für die Qualität des Bestandes bedeutet.
Hypermedial: Terminologiebstände werden miteinander und durch die Indexierung mit den Informationsbeständen vernetzt. Dabei wird die Konsistenz der Bestände insgesamt erhalten bzw. durch den Einsatz übergreifender konsistenzsichernder Verfahren verbessert.
3. Anfrageverfahren universell konzipieren:
Durch die einheitliche Repräsentation der Terminolgie- und Informationsbestände können Anfrageverfahren universell konzipiert werden. Eine universelle terminologische Anfragesprache ermöglicht den einheitlichen Zugriff auf alle Terminologieformate und damit auf den gesamten Informationsbestand.
Weitere Grundlagenforschung ist für den Bereich des terminologiebasierten Information Retrieval nötig. Durch Interdisziplinäre Forschungsprojekte wie ATLAS (ATLAS (1993)) können weitere Forschungsergebnisse erzielt werden.

7.5 Zukünftige Entwicklungsmöglichkeiten

Eine Erweiterung von StructuredQL lohnt sich in mehreren Bereichen:

Suche in einfach strukturierten Dokumenten:

- Hinzunahme weiterer Deskriptoren aus Z39.50-1995, STAS (1996) und ISO 8777
- Funktionen für Worthäufigkeiten
- Funktionen für Namensähnlichkeiten

Suche in link-strukturierten Dokumenten:

- Welche Nachfolger bis zur Stufe "5" und Suchbreite "10" hat das Dokument a ?
- Welche Nachfolger bis zur Stufe "5" und Suchbreite "10", die von "information" und "retrieval" handeln, hat das Dokument a ?
- Welche Dokumente rekursiv nur dann bis maximal zur Stufe "5" und Suchbreite "10" absteigend, wenn sie jeweils von "information" und "retrieval" handeln, sind Nachfolger des Dokuments a ?

Suche in geschachteltenen Dokumenten:

- Welche Dokumente und alle darin zitierten Dokumente handeln von "Information Retrieval"?
- Welche Dokumente enthalten letzter an Stelle (als hinterstes Objekt) ein Video ?
- Welche Beziehung besteht zwischen der Menge von Dokumenten M und dem Dokument a ?
- Bestimmte Anfragen in XQL bzw. XPath

Suche in Multimedia-Dokumenten (vgl. HyQ in DeRose Durand (1994)):

- 2 dimensionales Bild: Zeige den oberen 1/5 Teil. Zeige den Ausschnitt von Punkt (1,30) obere linke Ecke bis zum Punkt (300,400) untere rechte Ecke (Angaben auch in anderen Maßeinheiten möglich)
- 3 dimensionales Bild: Zeige den Ausschnitt [(1,1,1), (30,1,1), ...]

- Tonsequenz: Laß die Tonsequenz von der "3. Sekunde" bis zum Ende des Dokuments a hören.
- Bildsequenz: Zeige die ersten 10 Minuten des Dokuments a. Zeige die Passagen in dem Dokument a, in denen "Robert de Niro" mitspielt.
- Virtuelles Bild: Zeige den Hauptbahnhof in dem Dokument a.

Vage Suche:

- Welche Dokumente enthalten weit vorn ein Video in dem viel geschossen wird ?
- Welche Dokumente vom Typ "Video" enthalten Dokumente vom Typ "Ton", der höher ist als die Stimme von "Marilyn Monroe" ?

Statistische Suche:

- Auf welche Dokumente wird häufig zugegriffen?
- Wieviele Teile hat das größte Dokument?

Anfragen zum verwendeten Zeichensatz (Alphabet):

- Welche Dokumente enthalten kyrillische Zeichen (Zeichensatz ISO 8859-5) oder Zeichen des Zeichensatzes Unicode im Intervall i_1 bis i_2?

Entscheidungsfragen:

- Existiert das Dokument a ?

Je umfangreicher und komplexer die Syntax und Semantik einer Sprache aufgebaut ist, desto schwieriger wird ihre Erlern- und Beherrschbarkeit. Für verschiedene Benutzerklassen (Recherchelaie, geübter Rechercheur, Rechercheexperte, Informationsvermittler etc.) müssen deshalb Benutzer-Anfragemöglichkeiten unterschiedlichen Komplexitätsgrades zur Verfügung gestellt werden: Benutzeranfragesprache, Navigations- und Darstellungswerkzeuge, terminologiebasierte Unterstützung etc. Benutzeranfragen werden in die für den Benutzer verborgene universelle Anfragesprache übersetzt.

8 Erzielte Ergebnisse

These 1:
Man kann 3 Gruppen von strukturierten Dokumenten unterscheiden:
a) Einfach strukturierte Dokumente
b) Link-strukturierte Dokumente
c) Geschachtelte Dokumente
Es kann eine Anfragesprache entwickelt werden, mit der es möglich ist, alle 3 Gruppen universell anzufragen und deren jeweilige Vorteile zur verbesserten Recherche zu nutzen.

These 2:
Wenn zur Bestimmung eines Suchergebnisses die Anfrage an unterschiedlich mächtige Informationssysteme gesendet wird, kann in diesen keine Vereinigungsmenge, sondern nur die Schnittmenge des Suchergebnisses gebildet werden.

These 3:
Die Konsistenz von Terminologien wird durch die Einführung formaler Eigenschaften terminologischer Relationen verbessert.

These 4:
Ein System zur Rückkopplung zwischen Endnutzern, Informationsvermittlern, Terminologiepflegern, Indexierern und Autoren ermöglicht eine größere Übereinstimmung und Transparenz bei der Terminologiepflege und Indexierung.

9 Ausblick

In den Internet-Bibliotheken und -Buchhandlungen werden vermehrt strukturierte Dokumente mit Hypertextbeziehungen und multimedialen Elementen angeboten. Die Normierung dafür ist bereits weit fortgeschritten, zeigt aber ein noch starkes Innovationstempo. Der Zugriff auf Information ändert sich grundlegend. Die traditionelle Aufgabe des Information Retrieval bestand darin, relevante Dokumente wiederzufinden. In Zukunft besteht die Aufgabe darin, Dokumente, Teile von Dokumenten und in Beziehung stehende Dokumente und Dokumentteile zu lokalisieren. Weiterhin müssen sie nach Relevanz so geordnet werden, daß sie effizient erreicht und dem Benutzer adäquat präsentiert werden können. Es muß ein universeller Zugriff auf Dokumenttypen verschiedener Art geschaffen werden. Anfragemächtigkeit darf dabei nicht verloren gehen.

Betrachten wir die Bibliothek der Zukunft:
"Die Bibliothek muß leicht zugänglich sein, ihre Pforten müssen allen Mitgliedern der Gesellschaft offenstehen, so daß jeder sie frei benutzen kann, ohne Ansehen von Rasse, Hautfarbe, Nationalität, Alter, Geschlecht, Religion, Sprache, Personen- und Bildungsstand."
(Eco (1987) S. 38 zitiert die UNESCO)

Es herrscht Gründerzeit im Internet. Effizienzsteigerungen in großem Maße sind erreichbar. Arbeitskräfte werden frei, um die formal und inhaltlich neuartig strukturierten Dokumente qualitativ hochwertig zu sammeln, zu ordnen und den Benutzern zur Verfügung zu stellen. Entscheidend für ein verbessertes Retrieval ist die Kommunikation der Informationsanbieter mit den Informationssuchenden. Der Informationsanbieter stellt sich Fragen wie: Welche Informationswünsche hat der Suchende, was braucht er für seine Arbeit, für seine Freizeit, für seine Neugier, in welchen Zusammenhängen etc.? Wie geht er bei seiner Suche vor, welche Wege geht er dabei ? Der Informationssuchende stellt sich Fragen wie: Wie ist die Information angeordnet ? Welche Fragestellungen ergeben sich dadurch ? Stelle ich die Frage im richtigen Zusammenhang ? Frage ich zu eng oder zu weit ? Gehe ich bei meiner Frage von den richtigen Prämissen aus ? Sind meine Fragen in der richtigen Reihenfolge gestellt ? Wen spreche ich bei welcher Frage am besten an ?

In Zukunft besitzt der Mensch ein stoßfestes, wetterunempfindliches, tragbares, handliches Gerät, das es gestattet, kostenlos an und von jedem Ort zu publizieren, sich zu schulen, sich Medien aller Art zu beschaffen und präsentieren zu lassen, zu telefonieren, fernzusehen, Radio zu hören, Zeitung zu lesen, Nachrichten zu versenden, Güter zu beschaffen, Austellungen, Konzerte und Museen zu besuchen. In der virtuellen Bibliothek der Zukunft wird der Benutzer wie ein König behandelt. Das reichhaltige Sortiment ist ansprechend geordnet, durch Hinweise können Wege verkürzt und Abteilungen rasch gewechselt werden. Früher beschrittene Wege kann sich der Benutzer zeigen lassen. Die Bestände werden mehrdimensional bereitgestellt. Sachgebietskataloge allgemeiner und spezieller Art führen den Benutzer an günstige Positionen. Der Benutzer wird bei seiner Suche weder gestört noch mit Material überladen, er kann seine gefundenen Werke bequem lesen. Letztendlich wird er die Information zum Nutzen aller mehren. Wir profitieren gleichermaßen davon.

10 Anhang

10.1 Literatur

Abiteboul et.al. (1997). Abiteboul, S., Dallan, Q., McHugh, J. Widom, J., Wiener, J. The Lorel query language for semistructured data. Department of computer science, Stanford University, Stanford CA 94402, 1997.

Ackermann, Hilbert (1972). W. Ackermann, D. Hilbert. Grundzüge der theoretischen Logik (6. Auflage). Berlin, Springer 1972.

Afrati, Koutras (1990). Afrati. F. und C.D. Koutras. A hypertext model supporting query mechanisms. In Andrè, Rizk, Streitz 1990, S. 52-66.

André, Rizk, Streitz (1990). Rizk, A., Norbert Streitz und J. André. (eds.). Hypertext: concepts, systems and applications. Cambridge, University Press, 1990.

Andrews, Kappe, Maurer (1995). Andrews, K., Kappe, F., Maurer, H. The Hyper-G network information system. In: Journal of Universal Computer Science, Vol. 1, No. 4, 1995, S. 206-220. Oder unter: ftp://iicm.tu-graz.ac.at/pub/Hyper-G/doc/dms94.ps

Apple (1992). Apple Computer Inc. Inside Macintosh: Files. Addison-Wesley, 1992.

Ardö, Koch (1994). Ardö, A., Koch, T. Automatic classifikation of WAIS databases. Unter: www.ub2.lu.se/autoclass.html

Arocena, Mendelzon, Mihaila (1997). Arocena, G., Mendelzon, A. and Mihaila, G. Applications of a Web Query Language. In: Proceedings of the 6.th International WWW-Conference, Santa Clara, April 1997.

ATLAS (1993). ATLAS-Mitarbeiter. Methodenentwicklung für ein Archiv für Technik, Lebenswelt und Alltagssprache - Abschlußbericht. Technische Universität Berlin, 1993.

Atzeni, Masci et.al. (1998). Atzeni, P., Masci, A., Mecca, G., Merialdo, P. und G. Sindoni. From databases to Web-Bases: the ARANEUS experience. In: Technical Report 34-1998, Dipartimento di Informatica e Automazione, Universita di Roma Tre, Mai, 1998.

Baader et.al. (1990). Baader, F., Bürckert, H., Heinsohn, J., Hollunder, B., Müller, J., Nebel, B., Nutt, W., Profitlich, H. Terminological knowledge representation: a proposal for a terminological logic. Kaiserslautern, Deutsches Forschungszentrum für Künstliche Intelligenz, Technical Memo90-04, 1990.

Bakken (1995). Bakken, Stig Sæther. Ftp-Search 3.3. unter: ftpsearch.unit.no/.

Beck et.al. (1993). Beck, C., Finin, T., Fritzson, R., Genesereth, M., McKay, D., Pelavin, R., Shapiro, S., Weber, J. Specification of the KQML Agent-Communication Language.
Unter: www.cs.umbc.edu/kqml/kqmlspec/spec.html

Beeri, Kornatzky (1990). Beeri, C. und Y. Kornatzky. A logical query language for hypertext systems. In: André, Rizk, Streitz 1990, S. 67-80.

Bekavac (1996). Bekavac, B. Suchverfahren und Suchdienste des World Wide Web. In: Nachrichten für Dokumentation, Volume 47, No. 4, Juli, August 1996.

Boden et.al. (1994). Boden, K., Geenen, A., Kampermann, J., Scheller, M. Internet: Werkzeuge und Dienste von Archie bis World Wide Web. Berlin, Springer, 1994.

Böhm, Mengel, Muhr (1994). Böhm, Andreas, Andreas Mengel und Thomas Muhr (Hrsg.). Texte verstehen - Konzepte, Werkzeuge und Methoden. Konstanz, Schriftenreihe zur Informationswissenschaft, Konstanz, Universitätsverlag Konstanz, 1994.

Bowman et. al. (1995). Bowman, C., Danzig, P., Hardy, D., Manber, U., Schwartz, M., Wessels, D. Harvest: a scalable, customizable discovery and access system. Technical Report, Department of Computer Science, University of Colorado - Boulder, Mäz 1995. oder unter:
ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.Jour.ps.Z

Brachman, Schmolze (1985). Brachman, R. und Z. Schmolze. An overview on the KL-ONE knowledge representation system. In: Cognitive science, November 1985.
Brown (1989). Brown, P. Turning ideas into products: the GUIDE system. In: Proceedings of the first hypertext conference of ACM: Hypertext 87, ACM 1989.
Bush (1945). Bush, V. As we may think. The Atlantic Monthly, July 1945.
Camargo (1994). Camargo, W. The Harvest Broker. Pennsylvania State University, Department of Computer Science, Master of Science, Dezember 1994.
oder unter: harvest.transarc.com/afs/transarc.com/public/camargo/broker.ps
Carnap (1958). Carnap, R. Introduction to symbolic logic and its applications. New York, Dover Publications, 1958.
Cate (1992). Cate, Vincent. Alex - a global filesystem. School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
oder unter: www.funet.fi/FUNET/hamster/alex.intro
oder unter: ftp://alex.sp.cs.cmu.edu/www/alex.html.
Cattell (1994). Cattell R. (Hrsg.). The Object Database Standard: ODMG-93. San Francisco, Kaufman, 1994.
Conklin (1987). Conklin, J. Hypertext - An introduction and a survey. In: IEEE Computer Vol. 20 Nr. 9, S. 17-41.
Coulouris, Dollimore, Kindberg (1994). Coulouris, G., Dollimore, J., Kindberg, T. Distributed systems - concepts and design, second edition. Addison-Wesley, 1994.
Cruz, Mendelzon, Wood (1987). Cruz, I. Mendelzon, A., Wood, P. A graphical query language supporting recursion. Proceedings International Conference on Managemnet of Data 1987, S. 323-330.
Cruz, Mendelzon, Wood (1988). Cruz, I. Mendelzon, A., Wood, P. Recursive queries without recursion. Proceedings 2nd International Conference on Expert Database Systems 1988, S. 645-666.
Custer (1993). Custer, H. Inside Windows NT. Microsoft Press, 1993.
Czedik (1992). Czedik, Dorothea. Status Quo der Wiederverwendbarkeit von Wissensbasen. In: Künstliche Intelligenz, Zeitschrift der Gesellschaft für Informatik, S. 27-32.
Czyborra (2001). Czyborra, Roman. ISO 8859 Buchstabensuppe. unter: www.cs.tu-berlin.de/~czyborra/charsets/ .
Dalitz, Heyer (1995). Dalitz, W., Heyer, G. Hyper-G - Das Informationssystem der 2. Generation. Heidelberg, dpunkt-Verlag, Oktober 1995.
Daniel et.al. (1995). Ron Daniel, Jean Godby, Eric Miller, Stuart Weibel. OCLC/NCSA Metadata Workshop Report.
Unter: www.oclc.org:5046/conferences/metadata/dublin_core_report.html
DeRose, Durand (1994). DeRose, Steven; David Durand. Making hypermedia work: a user's guide to HyTime. Boston, Kluwer Academic Publishers, 1994.
Deutsch, Emtage (1992). Deutsch, P., A. Emtage. Archie - an electronic directory service for the Internet. Conference Proceedings USENIX 1992, pp. 93-110, San Francisco, CA, USA, Januar 1992. oder unter: ftp://ftp.th-darmstadt.de/pub/archie/doc/archie-usenix92-paper.[txt|ps].Z.
Dornseiff (1959). Dornseiff, Franz. Der Deutsche Wortschatz nach Sachgruppen. Berlin, De Gruyter, 1959.
DOS (1994). Benutzerhandbuch MS-DOS 6.22. Microsoft Corporation, Ireland, 1994.
Ewert, Umstätter (1997). Ewert, Gisela und Walter Umstätter. Lehrbuch der Bibliotheksverwaltung. Stuttgart, Hiersemann, 1997.
Eco (1987). Eco, Umberto. Die Bibliothek. Hanser Verlag, 1987.

Engelbart (1984). Engelbart, D. Authorship provisions in AUGMENT. In: Digest of papers, Compcon, 28th IEEE computer society international conference, San Francisco. Silver Spring, Computer Society press, 1984.
Felber, Budin (1989). Felber, Helmut und Gerhard Budin. Terminologie in Theorie und Praxis. Tübingen, Narr, 1989.
Ferguson (1994). Ferguson, George. Xarchie Version 2.0.10: X11 browser interface to Archie. unter: cs.rochester.edu/u/ferguson oder Email: ferguson@cs.rochester.edu, Juli 1994.
Fernandez, Popa, Suciu (1997). Mary Fernandez , Lucian Popa , Dan Suciu. A structure based approach to querying semistructured data. In: Proceedings of the Workshop on Database Programming Languages , 1997.
Flanagan (1996). Flanagan, David. Java in a Nutshell (1. Ausgabe in deutscher Sprache). O'Reilly, 1996.
Foster (1994a). Foster, Steven. Installing the veronica server and data - version 0.6.5f. Juni, 1994.
unter: gopher://futique.scs.unr.edu:70/00/veronica/About/Server/INSTALL.
Foster (1994b). Foster, Steven. How to compose Veronica queries. Juni, 1994.
unter: gopher://veronica.scs.unr.edu:70/00/veronica/how-to-query-veronica.
Fuhr (1991). Fuhr, Norbert (Hrsg.). Information Retrieval. Berlin, Informatik Fachberichte Nr. 289. Berlin et.al, Springer Verlag, 1991.
Fuhr (1995). Fuhr, Norbert. Modelling Hypermedia Retrieval in Datalog. In: Proceedings HIM'95. Universitätsverlag Konstanz, Konstanz, April 1995.
Fuhr et. al. (1995). Fuhr, N., Huynh, T., Pfeifer, U. Searching structured documents with the enhanced retrieval functionality of freeWAIS-sf and SFgate. Computer Networks and ISDN Systems 27 (6), 1995, pp 1027-1036. oder unter:
www.igd.fhg.de/www/www95/papers/47/fwsf/fwsf.html
Garcia-Molina, Paepcke (1996). Garcia-Molina, H., Paepcke, A. Proposal for a I**3 client server protocol. Technical Report, September 1996.
Garcia-Molina, Hammert et.al. (1995). Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y.Ullman, J. und J. Widom. Integrating and accessing heterogeneous information sources in TSIMMIS. In Proceedings of the AAAI Symposium on Information Gathering, pp. 61-64, Stanford, California, März 1995.
Gilster (1995). Gilster, P. Suchen und Finden im Internet. Wien, Hanser Verlag, 1995.
Goldfarb, Prescod 2001. Golfarb, Charles and Paul Prescod. The XML-Handbook (3rd ed.), Prentice Hall, 2001.
Gövert, Pfeifer (1996). Gövert, N., Pfeifer, U. SFgate: The WWW Gateway for freeWAIS-sf; Edition 0.1, for SFgate 5.0. Mai 1996. Unter:
ls6.informatik.uni-dortmund.de/ir/projects/SFgate/SFgate.html
Gövert (1996a). Gövert, N. Information Retrieval in vernetzten heterogenen Datenbanken. 1996.
unter: ls6.informatik.uni-dortmund.de/ir/reports/96/Goevert-96.html und unter:
ls6.informatik.uni-dortmund.de/ir/projects/SFgate/heterogeneous.html
Gövert (1996b). Gövert, N. SFgate 5.019: Searching for databases.
Unter: ls6.informatik.uni-dortmund.de/ir/projects/SFgate/index.html
Guha, Lenat (1990a). Guha, L. und Lenat, Douglas. Building large knowledge-based systems - representation and inference in the cyc project. Addison-Wesley 1990.
Guha, Lenat (1990b). Guha, R. und Lenat, Douglas. CycL: The Cyc Representation Language Part 4. Technischer Bericht, ACT-CYC 154-90, 1990. anzufordern bei library@mcc.com oder unter:
www.cyc.com/tech-reports/act-cyc-154-90/act-cyc-154-90.html

Gulbins, Obermayr (1995). Gulbins, J. und K. Obermayr. UNIX - System V.4. Begriffe, Konzepte, Kommandos, Schnittstellen. Berlin et.al., Springer Verlag, 1995.
Hausmann (1985). Hausmann, Franz Josef. Lexikographie. In: Schwarze, Wunderlich (Hrsg.). Handbuch der Lexikologie. Königstein/Ts, Athenäum, 1985.
Handschuch (1995). Handschuch, T. Solaris 2 Systemadministration. Thomson Publishing, 1995.
Hartlep (1996). Hartlep, Frank. Realisierung eines Werkzeugs zur Hypertextualiserung von Thesauri. Diplomarbeit, Technische Universität Berlin, FB Informatik, Fachgebiet WBS, 1996. oder das System unter: www.josef-willenborg.de/thesaurus/Overview.html
Huttel (1995). Huttel, K. OS/2 Warp Version 3. Thomson Publishing, 1995.
iX 5/1996. iX - Multiuser Multitasking Magazin. Heise Verlag, Mai 1996.
Jacobs, Shea (1996). Jacobs, N., Shea, R. The Role of Java in InfoSleuth: Agent-based Exploitation of Heterogeneous Information Resources. Microeclectronics and Computer Technology Corporation (MCC), 3500 Balcones Center Dr. Austin, Texas 78759, 1996.
oder unter: www.mcc.com/projects/infosleuth/papers/intranet-java.html
Jennings, Wooldridge (1995). N. Jennings, Wooldridge, M. Intelligent Agents: Theory and Practice. In: Knowledge Engineering Review 10(2), 1995.
Jones (1994). Jones, Rhett "Jonzy" (e-mail: jonzy@cc.utah.edu). Jughead Version 1.0.4: Jonzy's Universal Gopher Hierarchy Excavation And Display. University of Utah Computer Center, 1994. siehe auch unter: gopher://gopher.cc.utah.edu/
Kappe (1993). Kappe, F. Hyper-G: a distributed hypermedia system. In: Proceedings INET 93, San Francisco, California, August 1993.
Oder unter: ftp://iicm.tu-graz.ac.at/pub/Hyper-G/doc/inet93.ps
Kappe (1996), Kappe, F. Hyper-G Text Format (HTF) Version 2.13.
Unter:ftp://iicm.tu-graz.ac.at/pub/Hyper-G/doc/HTF.ps
Kappe, Pani (1996). Kappe, F., Pani, G. Hyper-G Client-Server Protocol (HG-CSP), Version 7.17. In: Maurer (1996) Anlage F oder unter:
ftp://ftp.iicm.tu-graz.ac.at/pub/Hyper-G/papers/HG-CSP-7.17.hif.gz
Khalidi et.al. (1993). Khalidi, Y., P. Madany, M. Nelson. The Spring File System. Sun Microsystems Laboratories Technical Report 93-10, März 1993.
Kleinberg, Kumar et.al. (1999). Kleinberg, J., Kumar, R., Raghavan, P., Rajagopalan, S. und A. Tomkins. The web as a graph: measurements, models, and methods. In: Proccedings of the Internation Conference on Combinatorics and Computing, 1999.
Koch (1995). Koch, T. Searching the Web - systematic overview over indexes. In: Hobohm, Wätjen (Hrsg.) Wissen in elektronischen Netzwerken - eine Auswahl von Vorträgen der Gesellschaft für Klassifikation Basel 1995, Bibliotheks- und Informationssystem der Universität Oldenburg, 1995.
Kofler (1995). Kofler, M. Linux - Installation, Konfiguration, Anwendung. Addison Wesley, 1995.
Konopnicki, Shmueli (1998). Konopnicki, D., Shmueli, O. Information gathering in the World Wide Web: The W3QL query language and the W3QS System. In: ACM Transactions on Database Systems, Vol. 23, No 4, December 1998, pp 369-410.
Konrad (1976). Konrad, Erhard. Formale Semantik von Datenbanksprachen. Dissertation an der TU Berlin, Fachbereich Informatik.
Konrad, Reiner (1985). Konrad, Erhard, Ulrike Reiner. Eine semantische Analyse der Suchkomponente des Information-Retrieval-Systems GRIPS. Technische Universität Berlin, Fachbereich Informatik, Fachgebiet Computergestützte Informationssysteme, LIVE-Bericht Nr. 2/85, 1985.

Konrad (1986). Konrad, E. Informationssysteme I. Skript zur gleichnamigen Lehrveranstaltung, Fachbereich Informatik, TU Berlin, 1986.
Konrad (1992). Konrad, Erhard. Zur Effektivitätsbewertung von Information-Retrieval-Systemen. In: Experimentielles und Praktisches Information Retrieval (ed. Kuhlen). Schriften zur Informationswissenschaft Bd.III. Universitätsbibliothek Konstanz, S. 119-130.
Kreitzberg, Shneiderman (1988). Kreitzberg, C., Shneiderman, B. Restructuring knowledge for an electronic encyclopedia. In: Proceedings International Ergonomics Association 10th Congress 31, vol. 2, (Sydney, Australia, Aug. 1-5, 1988) 615-620.
Kuhlen (1991). Kuhlen, Rainer. Hypertext, ein nichtlineares Medium zwischen Buch und Wissensbank. Berlin, Heidelberg, Springer Verlag, 1991.
Kumar, Raghavan et.al. (1999). Kumar, R., Raghavan, P., Rajagopalan, S. und A. Tomkins. Extracting large-scale knowledge bases from the web. In: IEEE International Conference on Very Large Databases (VLDB), Edinburgh, 1999.
Lakshmanan, Sadri, Subramanian (1996). Lakshmanan, L., Sadri, F. und I. Subramanian. A declarative language for querying and restructuring the web. In: Post-ICDE IEEE Workshop on Research Issues in Data Engineering (RIDE-NDS'96), New Orleans, Februar 1996.
Lapp, Robie, Schach (1998). Lapp, J., Robie, J. und D. Schach. XML Query Language (XQL). Unter: www.w3.org/TandS/QL/QL98/pp/xql.html
Lauer, Scholz (1996). Novell Netware 4.1. Mark&Technik Buch- und Software-Verlag, 1996.
Lea (1995). Lea, Iain (iain.lea@erlm.siemens.de). Tin - a netnews reader (Version 1.2.) . Manual Pages, 1995.
Liebe (1995). Liebe, Andreas. Wer suchet, Geschichte und Funktionsweise des Internet-Dienstes. In: iX, März 1995, S. 168-175.
Lyons (1980). Semantik (Band 1 und 2). Beck, München, 1980.
Manber, Wu (1992). Manber, U., Wu, S. Agrep - a fast approximate pattern-matching tool. Usenix 1992 Technical Conference, San Francisco, pp. 153-162.
Manber, Wu (1994). Manber, U., Wu, S. GLIMPSE: A Tool to Search Through Entire File Systems. Proceedings of the USENIX Winter Conference, pp. 23-32, San Francisco, California, Januar 1994. oder unter: ftp://cs.arizona.edu/reports/1993/TR93-34.ps.Z
Manber (1996). Manber, U. Glimpse Help Pages (GLIMPSE Version 3.5.). April 1996. Unter: glimpse.cs.arizona.edu/glimpsehelp.html
Masermann, Vossen (1998). Masermann, U. und G. Vossen. Suchmaschinen und Anfragen im World Wide Web. In: Informatik Spektrum 21, pp. 9-15, 1998.
Maurer (1996). Maurer, Hermann. HYPERWAVE - The Next Generation Web Solution. Addison Wesley, 1996.
Mendelzon, Mihaila, Milo (1997). Mendelzon, A., Mihaila G. und T. Milo. Querying the World Wide Web. In: Journal of Digital Libraries 1 (1), pp 68-88, 1997.
Middendorf, Singer, Strobel (1996). Middendorf, Stefan, Reiner Singer und Stefan Strobel. Java - Programmierhandbuch und Referenz. Heidelberg, Dpunkt - Verlag, 1996.
Miller (1995). Miller, George. WordNet: A Lexical database for English. Communications of the ACM. November 1995, 39-41.
Nelson (1993). Nelson, Ted. Literary Machines. Mindful Press, California/USA, 1993.
Niedermair (1995. Niedermair, Klaus. Hyperkatalog mit BIBOS-Daten - am Beispiel der WWW-Site info.uibk.ac.at/c108/pub_uibk. In: ONLINE-MITTEILUNGEN der Oesterreichischen Online-Benutzergruppe, Nr.52, Juni 1995, ISSN 1015-1869. oder unter: info.uibk.ac.at/c108/c10806/voeb/om52.html#Klaus Niedermair52
Nielsen (1996). Nielsen, Jakob. Multimedia, Hypertext und Internet. Grundlagen und Praxis des elektronischen Publizierens. Braunschweig, Vieweg Verlag, 1996.
Nilsson (1982). Nilsson, N. Principles of Artificial Intelligence. Springer 1982.

O'Donell (1994). O'Donell, S. Programming for the world - a guide to internationalization. Prentice Hall, New Jersey, 1994.
Papakonstantinou, Garcia-Molina, Widom (1995). Papakonstantinou, Y.,Garcia-Molina, H. und Widom, J. Object exchange across heterogeneous information sources. In: IEEE International Conference on Data Engineering, pp 251-260, Taipeh, Taiwan, März 1995.
Petkovic (1995). Petkovic, Dusan. Informix 6.0/7.1. Addison-Wesley, 1995.
Pfeifer (1995. Pfeifer, U. FreeWAIS-sf - the enhanced freeWAIS distribution. Oktober 1995. unter: ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/index.html
Pu, Schwartz (1994). Pu, Calton und Michael Schwartz. Applying an Information Gathering Architecture to Netfind: A White Pages Tool for a Changing and Growing Internet. IEEE/ACM Transactions on Networking, 2(5), 426, October 1994. Oder unter:
ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Netfind.Gathering.ps.Z
Quass, Rajaraman et. al. (1995). Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., und J. Widom. Querying semistructured heterogeneous information. In: International Conference on Deductive and Object-Riented Databases, 1995.
Ray, Ray, Seitzer (1997). Ray, D., E. Ray, R. Seitzer. The AltaVista Search Revolution. McGraw Hill, 1996.
Reiner (1991). Reiner, Ulrike. Anfragesprachen für Informationssysteme. Reihe Informationswissenschaft der DGD, Bd.I. Frankfurt a.Main, Deutsche Gesellschaft für Dokumentation, 1991.
Reiner (1994). Reiner, Ulrike. Anfragesprachen für Textsuchsysteme. In: Böhm, Mengel, Muhr 1994.
Salton (1975). Salton, G. Dynamic information and library processing. Englewood, Prentice Hall, 1975.
Salton, McGill (1983). Salton, Gerard und Michael McGill. Introduction to modern Information Retrieval. McGraw-Hill, 1983.
Schmidt, Ströhlein (1993). Schmidt, G. und T. Ströhlein. Relations and Graphs. Berlin, Heidelberg, New York, Springer Verlag, 1993.
Schönfeldt (1994). Schönfeldt, René. Mathematische Eigenschaften für Thesaurusrelationen. In: Nachrichten für Dokumentation 45, Nr. 4, S. 203-212, 1994.
Shannon, Weaver (1963). C. E. Shannon and W. Weaver. The Mathematical Theory of Communication. Urbana, Illinois, University of Illinois Press, 1949, republished in paperback 1963.
Soergel (1974). Indexing languages and thesauri: construction and maintenance. Los Angeles, Melville Publishing Company, 1974.
Stallings (1995. W. Stallings. Sicherheit im Datennetz. München, Prentice Hall, 1995.
STAS (1996). The Scientific and Technical Attribute and Element Set (STAS). CNIDR - Clearinghouse for Networked Information Discovery and Retrieval, Research Triangle Park, North Carolina, 1996 oder unter: stas.cnidr.org/.
Stern (1995). Stern, H. Verwaltung von UNIX-Netzwerken mit NFS und NIS. O'Reilly, 1995.
Storm (1995. Storm, K. (storm@texas.dk). Netnews 6.5 - an efficient netnews interface (manual pages). Texas Instruments, Denmark, 1995.
Tanenbaum (1995). Tanenbaum, A. Moderen Betriebssysteme, 2. Auflage. Hanser Verlag, 1995.
Umstätter (1998). Umstätter, Walter. Die Messung von Wissen. In: Nachrichten für Dokumentation 49 (4) S. 221-224, 1998.
Volpert (1985). Volpert, W. Zauberlehrlinge - Die gefährliche Liebe zum Computer. Beltz Verlag, 1985.
Willenborg (1991a). Willenborg, Josef. PfleSaurus - ein System zur Erstellung und Weiterentwicklung von Thesauri. Unveröff. Diplomarbeit an der TU Berlin, Institut für Angewandte Informatik.

Willenborg (1991b). Willenborg, Josef. ATLAS-PfleSaurus: Ein objektorientiertes System zur Unterstützung der Thesauruspflege. In: Fuhr (1991) S. 51-63.
Willenborg (1993). Willenborg, Josef. Hypermediabasierte Terminologie- und Wörterbuchpflege. In: Neubauer, Wolfram (Hrsg.). Deutscher Dokumentartag 1992. Proceedings. Deutsche Gesellschaft für Dokumentation, Frankfurt am Main, 379-410.
Willenborg (1994a). Willenborg, Josef. PfleSaurus - ein Werkzeug zur Thesauruspflege. In: Böhm, Mengel, Muhr (1994).
Willenborg (1994b). Willenborg, Josef. Terminologiebasiertes Hypertext Retrieval. In: Böhm, Mengel, Muhr (1994).
Windows (1992). Microsoft Windows 3.1 Handbuch. Microsoft Corporation, 1992.
Zloof (1976). Zloof, M. Query by example: operations on the transitive closure. IBM Research 5526 (revised), Yorktown Heights, IBM Research Center.

10.2 URL von Personen, Organisationen und Produkten

Die Jahreszahlen am Ende jeder Angabe beziehen sich auf den letzten erfolgreich durchgeführten Zugriff.

Aliweb (2000). Welcome to ALIWEB. Unter: aliweb.emnet.co.uk, Oktober 2000.
AltaVista (2000). Alta Vista: Welcome. Unter: www.altavista.com/, Oktober 2000.
Apollo (1997). Apollo Advertising. Unter: apollo.co.uk/, März 1997.
Araneus (2001). ARANEUS HOME PAGE. Unter: www.dia.uniroma3.it/Araneus, Februar 2001.
Autonomy (2001). Autonomy - Automating the Digital Economy. Unter: www.autonomy.com/, Januar 2001.
Bigfoot (2001). Bigfoot, the global e-mail directory for the online community.
Unter: www.bigfoot.com., Februar 2001.
Bunyip (1996). Bunyip Information Systems, 310 St-Catherine St. West, Suite 202, Montreal, Quebec H2X 2AI, Telefon: 514-875-8611, E-mail: info@bunyip.com unter:
www.bunyip.com
Chemie (1997). WWW Server FB Chemie FU Berlin - Search for Servers in WWW (Welcome Pages). Unter:
www.chemie.fu-berlin.de/cgi-bin/srch.cgi/outerspace/www-servers, März 1997.
CIFS (1997). Common Internet File System Resource Center. Unter:
www.cifs.com, März 1997.
Clever (2001). The Clever Project. Unter: www.almaden.ibm.com/cs/k53/clever.html, Februar 2001
CNIDR (1996). Center for Networked Information Discovery and Retrieval.
Unter: www.cnidr.org/
Cyc (2001). Cyc Corporation. Upper Cyc Ontology.
Unter: www.cyc.com/cyc-2-1/cover.html. Februar 2001.
Cyveillance (2001). Unter: http://cyveillance.com/us/newsroom/pressr/000710.asp, Januar 2001
DejaNews (2000a). Dejanews: The source for Internet Newsgroups - Homepage.
unter: www.dejanews.com, August 2000.
DejaNews (2000b). Deja.com: Corporate Site. Unter: www.dejanews.com/corp/about.shtml, August 2000.
DeTeMedien (2001a). DeTeMedien: Teleauskunft 1188.
Unter: www.teleauskunft1188.de/, Februar 2001.
DeTeMedien (2001b). Gelbe Seiten / Yellow pages. DeTeMedien der Deutschen Telekom.

Unter: www.gelbe-seiten.de, Februar 2001.
DIB (2001). Deutsches Internet Branchenbuch. Unter: www.branchenbuch.de/, Februar 2001.
DIMDI (2001). Deutsches Institut für medizinische Dokumentation und Information. Unter: www.dimdi.de/, Februar 2001.
DublinCore (2001). Dublin Core metadata element set: reference description. unter: purl.org/metadata/dublin_core_elements, Februar 2001
Excite (2000). My Excite start page. Unter: www.excite.com/, Oktober 2000.
Explorer (2001). Microsoft Internet Explorer ©2001 Microsoft Corporation. Unter: www.microsoft.com/
Fireball 2000. Fireball Express Suche. Unter: www.fireball.de, Oktober 2000.
Four11 (1997). Four11 - the internet white pages. Unter: www.Four11.com, März 1997.
FreeWAIS (2001). Jim Fullton, Kevin Gamiel, Archie Warnock, et.al. FreeWAIS, Version 0.5. Unter: ftp://ftp.cnidr.org/pub/NIDR.tools/freewais/ , Februar 2001.
Google (2000). Google. Unter: www.google.com, Oktober 2000.
IAF (2001). Internet address finder. Unter: www.iaf.net/, Februar 2001.
Harvest (2000). Harvest Information Discovery and Access System.
Unter: http://www.tardis.ed.ac.uk/harvest/
oder harvest.transarc.com/harvest/, Oktober 2000.
HBZ (2001a). Hochschulbibliothekszentrum des Landes Nordrhein-Westfalen.
Unter: www.hbz-nrw.de/, Febraur 2001
HBZ (1997b). Bibliotheks-OPAC's und -informationsseiten.
Unter: www.hbz-nrw.de/novell/etc/hartges/opac.htm, März 1997.
HBZ (1997c). Hochschulbibliothekszentrum Nordrhein-Westfalen: Personennamen.
Unter: www.hbz-nrw.de/novell/etc/hartges/person.htm, März 1997.
HBZ (1997d). Buchhandel (Barsortimente, SortimenteVIBs).
Unter: www.hbz-nrw.de/novell/etc/hartges/buchhand.htm, März 1997.
HBZ (1997e). Verlage. Unter: www.hbz-nrw.de/novell/etc/hartges/verlag.htm, März 1997.
Horlacher (2001). SAFT - Simple Asynchronous File Transfer.
Unter: www.belwue.de/belwue/software/saft/saft.html, Februar 2001.
Hypercard (1997). HyperCard Version 2.3. © 1997 Apple Computer, Inc. Unter: product.info.apple.com/productinfo/datasheets/as/hypercard2.3.html
Hyperkatalog Innsbruck (2001). Hyperkatalog der Publikationen an der Universität Innsbruck. Unter: info.uibk.ac.at/c108/pub_uibk/, Februar 2001.
HyTelnet (1997). HyTelnet on the World Wide Web. Unter: library.usask.ca/hytelnet/, März 1997.
IFS (2000). Oracle Internet File System. Unter: www.oracle.com/database/options/ifs/
InfoMagnet (2001). InfoMagnet. Unter: www.compassware.com/products/, Januar 2001.
Infoseek (2000). Infoseek. Unter: www.infoseek.de, Oktober 2000.
Inktomi (2000). Unter: www.ultraseek.com, Oktober 2000.
ISC (2000). Internet Software Consortium.
Unter: www.isc.org/ds/WWW-200001/report.html, August 2000.
Kolibri (2000). Kolibri Suchmaschine. Unter: www.kolibri.de, Oktober 2000.
Koster (2001a). Koster, M. WWW Robot Frequently Asked Questions.
Unter: info.webcrawler.com/mak/projects/robots/faq.html, Februar 2001.
Koster (2001b). Koster, M. The Web Robots Database.
Unter: info.webcrawler.com/mak/projects/robots/active.html, Februar 2001.
Linkstar (1997). LinkStar business directory. Unter: www.linkstar.com/, März 1997.
Liszt (2000). Liszt, the mailing list directory. Unter: www.liszt.com/, August 2000.

Lycos (2000). Lycos: Ihr persönlicher Internet Guide Unter: www.lycos.de, Oktober 2000.
MAB (1997). Die Deutsche Bibliothek - MAB.
Unter: www.ddb.de/profil/zsarbeit/stabil/mab.htm, März 1997.
Messenger (2001). STN's Retrieval Language MESSENGER.
Unter: www.FIZ-Karlsruhe.DE/stn/messenger/mc-toc.html, Februar 2001.
Metacrawler (2000). Metcrawler. Unter: www.metacrawler.com/, Oktober 2000.
Mids (2000). Matrix Information and Directory Services, Inc. (MIDS). Unter: www.mids.org/works.html , August 2000.
Netcraft (2000). The Netcraft Web Server Survey. Unter: www.netcraft.co.uk/Survey/ , August 2000.
Netfind (1997). Internic Netfind Search (mit 1,2 Millionen Rechnereinträgen). Unter: www.internic.net/wp/netfind.html, März 1997.
Netscape (2001). Netscape Browser Central, Netscape Version 6.
Unter: home.netscape.com/browsers/index.html, Februar 2001.
NlightN (1997). NlightN Home Page! Unter: www.nlightn.com/, März 1997.
Nua (2000a). Nua Internet How Many Online. Unter:
www.nua.ie/surveys/how_many_online/index.html, August 2000.
Nua (2000b). Nua Internet Surveys: Graphs & Charts - 2000. Unter:
www.nua.ie/surveys/analysis/graphs_charts/1999graphs/b2c_spend.html, August 2000.
OCLC (2001). OCLC Online Computer Library Center, Inc. Unter: www.oclc.org/, Februar 2001.
OpenText (2000). OpenText Corporation. Unter: www.opentext.com, Oktober 2000.
Oracle (2000a). Oracle interMedia. Unter: www.oracle.com/intermedia/ , April 2000.
Oracle (2000b). Products interMedia. Unter: technet.oracle.com/products/intermedia/ , April 2000.
Oracle (2001). Oracle9i AS Portal. Unter: http://technet.oracle.com/products/iportal/, Februar 2001.
RBSE (1997). RBSE Spider. Unter: rbse.jsc.nasa.gov/Spider/query.html, März 1997.
Scott (1997). Peter Scotts Homepage. Unter: duke.usask.ca/~scottp/, März 1997.
SearchEngineWatch 2000. Search Engine Sizes. Unter:
www.searchenginewatch.com/reports/sizes.html, August 2000.
SearchEngineWatch (2001). Unter: www.seachenginewatch.com, Januar 2001.
STN (1997). STN International: databases in science and technology. Unter:
www.fiz-karlsruhe.de/stn.html, März 1997.
Switchboard (2001). Welcome to Switchboard - the people and business directory. Unter:
www.switchboard.com/, Februar 2001.
Thesaurus Guide (1985). Thesaurus guide. Luxembourg, Office for Official Publications of the European Communities.
Thesaurus Graphics (2001). LCTGM: The Library of Congress: Thesaurus for Graphic Material. Unter: www.loc.gov/rr/print/tgm1/, Februar 2001.
Thinking Machines (1992). WAIS, Version 0.8 b5 unter:
ftp://wais.com/pub/freeware/unix-src/wais-8-b5.1.tar.Z
Tuchman (1993). Tuchman, Allan (e-mail: a-tuchman@uiuc.edu). Xgopher - Version 1.3. Computing and Communications Services Office (CCSO), University of Illinois at Urbana-Champaign, 1304 W. Springfield Ave., Urbana, Illinois, 61801, USA, 1993.
UMBC (2001). UMBC AgentWeb. Unter: agents.umbc.edu, Januar 2001.
Unicode (2001). Unicode Homepage. Unter: www.unicode.org/ , Februar 2001.
UNIMARC (1997). Die Deutsche Bibliothek - UNIMARC.

Unter: www.ddb.de/profil/zsarbeit/stabil/unimare.htm, März 1997.
W3QS (1999). Unter: www.cs.technion.ac.il/~W3QS, August 1999.
Waissearch (1992). Waissearch(1), Manual Pages von Thinking Machines Corp. Brewster@think.com. Februar 1992.
Waisq (1992). Waisq(1), Manual Pages von Thinking Machines Corp. Brewster@think.com. Februar 1992.
Webcrawler (2000). WebCrawler. Unter: www.webcrawler.com, Oktober 2000.
WebNFS (2001). WebNFS. Unter: www.sun.com/webnfs/, Februar 2001.
WebSQL (1999). Unter: www.cs.toronto.edu/~websql/, August 1999.
WebWhacker (2001). Web Whacker 2000. Unter: www.webwhacker.com/ Januar 2001.
Word Net (1997). WWW interface to WordNet 1.5. unter: www.cogsci.princeton.edu/~wn/w3wn.html, März 1997.
X.500-WWW (1997). X.500 White Pages Directory. Unter: ds2.internic.net:8888/ , März 1997.
Xwais (1992). Xwais(1), Manual Pages von Thinking Machines Corp. Brewster@think.com. Februar 1992.
Yahoo (2000). Yahoo! Unter: www.yahoo.com/, Oktober 2000.
Yellow Pages (1996). World Wide Yellow Pages. Unter: www.yellow.com/, 1996.

10.3 Normen

National

DIN-Index. Deutsches Institut für Normung e.V. Unter: www.din.de
DIN 1463 Teil 1: (Nov. 1987) Erstellung und Weiterentwicklung von Thesauri; Einsprachige Thesauri
DIN 1463 Teil 2: (Entwurf Dez. 1988) Erstellung und Weiterentwicklung von Thesauri; Mehrsprachige Thesauri
DIN 2330: (März 1979) Begriffe und Benennungen; Allgemeine Grundsätze
DIN 2331: Begriffsysteme und ihre Darstellung
DIN 2332: Benennen international übereinstimmender Begriffe
DIN 2333: (Dez. 1987) Fachwörterbücher, Stufen der Ausarbeitung
DIN 2335: (Okt. 1986) Sprachenzeichen
DIN 2336: (März 1979) Lexikographische Zeichen für manuell erstellte Fachwörterbücher
DIN 2339 Teil 1: (Mai 1987) Ausarbeitung und Gestaltung von Veröffentlichungen mit terminologischen Festlegungen
DIN 2339 Teil 2: (Entwurf August 1986) Ausarbeitung und Gestaltung von Veröffentlichungen mit terminologischen Festlegungen
DIN 2341 Teil 1: (Entwurf Okt. 1986) Format für den maschinellen Austausch terminologischer/lexikographischer Daten - MATER; Kategorienkatalog
DIN 2342 Teil 1: (Entwurf Dez. 1986) Begriffe der Terminologielehre
DIN 31623 Teil 1 - 3: (September 1988) Indexierung zur inhaltlichen Erschließung von Dokumenten
DIN 66315: (1992) Database Language SQL.
RAK. Regeln für die alphabetische Katalogisierung. Wiesbaden, Reichert, 1977.
RSWK. Regeln für den Schlagwortkatalog. RSWK/bearbeitet von der Kommission des DBI für Sacherschließung. Berlin, Deutsches Bibliotheksinstitut, 1991.
Schlagwortnormdatei. Schlagwortnormdatei (SWD). bearb. von der Deutschen Bibliothek, Frankfurt am Main, Deutsche Bibliothek, 1993.

American National Standards Institute (ANSI)

ANSI-Index. Searching the ANSI Catalog. American National Standards Institute (ANSI). März 1996. Unter: www.ansi.org/cat_c.html

Z39.50-1995. Z39.50. Information Retrieval: Application Service Definition and Protocol Specification for Open Systems Interconnection. ANSI/NISO, July 1995.

International

Requests for Comments (RFC)

RFC-Index. Request for comments. Internet Society (ISOC), IETF Secretariat - Corporation for National Research Initiatives. Unter: www.ietf.org/ oder unter www.faqs.org/rfcs/ .
RFC 742. K. Harrenstien. Name/Finger. Dezember 1977.
Unter: www.faqs.org/rfcs/rfc742.html
RFC 791. Postel, J. Internet Protocol, Darpa Internet Program, Protocol specification. September 1981. Unter: www.faqs.org/rfcs/rfc791.html
RFC 793. Transmission Control Protocol, Darpa Internet Program, Protocol specification. September 1981. Unter: www.faqs.org/rfcs/rfc793.html
RFC 812. K. Harrenstien und V. White. Nicname/Whois. 1982.
Unter: www.faqs.org/rfcs/rfc812.html
RFC 821. J. Postel. Simple mail transfer protocol. 1982.
Unter: www.faqs.org/rfcs/rfc821.html
RFC 854. J. Postel, J. Reynolds. Telnet protocol specification. Mai 1983.
Unter: www.faqs.org/rfcs/rfc854.html
RFC 954. K. Harrenstien, M. Stahl, E. Feinler. Nicname/Whois. Oktober 1985.
Unter: www.faqs.org/rfcs/rfc954.html
RFC 959. J. Postel, J. Reynolds. File transfer protocol (ftp). Oktober 1985.
Unter: www.faqs.org/rfcs/rfc959.html
RFC 977. B. Kantor, P. Lapsley. Network News Transfer Protocol - A Proposed Standard for the Stream-Based Transmission of News. Februar 1986.
Unter: www.faqs.org/rfcs/rfc977.html
RFC 1014. Sun Microsystems, Inc. XDR: External Data Representation Standard. Juni 1987.
Unter: www.faqs.org/rfcs/rfc1014.html
RFC 1034. P. Mockapetris. Domain names - concepts and facilities. November 1987. Unter: www.faqs.org/rfcs/rfc1034.html
RFC 1036. M. Horton, R. Adams. Standard for Interchange of USENET Messages. December 1987. Unter: www.faqs.org/rfcs/rfc1034.html
RFC 1057. Sun Microsystems, Inc. RPC: Remote Procedure Call - Protocol Specification Version 2. Juni 1988. Unter: www.faqs.org/rfcs/rfc1057.html
RFC 1094. Sun Microsystems, Inc. NFS: Network File System Protocol Specification.März 1989. Unter: www.faqs.org/rfcs/rfc1094.html
RFC 1288. D. Zimmerman. The Finger User Information Protocol. Dezember 1991.
Unter: www.faqs.org/rfcs/rfc1288.html
RFC 1292. R. Lang, R. Wright. A Catalog of Available X.500 Implementations. Lawrence Berkeley Laboratory (Editors), Januar 1992.
RFC 1308. C. Weider, J. Reynolds. Executive Introduction to Directory Services: Using the X.500 Protocol. März 1992.
RFC 1309. C. Weider, J. Reynolds, S. Heker. Technical Overview of Directory Services: Using the X.500 Protocol. März 1992.

RFC 1436. F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. John, D. Torrey, B. Alberti. The Internet Gopher Protocol (a distributed document search and retrieval protocol). 1993. Unter:www.fagg.uni-lj.si/MIRROR/rfc/rfc/rfc1436.html oder: www.faqs.org/rfcs/rfc1436.html

RFC 1521. N. Borenstein, Bellcore, N. Freed, Innosoft. MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. September 1993. Unter: www.faqs.org/rfcs/rfc1521.html

RFC 1522. K. Moore. MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text. September 1993. Unter: www.faqs.org/rfcs/rfc1522.html

RFC 1625. St. Pierre, M, Fullton, J., Gamiel, K., Goldman, J., Kahle, B., Kunze, J., Morris, H., und F. Schiettecatte. WAIS over Z39.50-1988. WAIS, Inc., CNIDR, Thinking Machines Corp., UC Berkeley, FS Consulting, June 1994. Unter: www.faqs.org/rfcs/rfc1625.html

RFC 1630. T. Berners-Lee. Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web. Juni 1994. Unter: www.faqs.org/rfcs/rfc1630.html

RFC 1714. Kosters, M., Williamson, S. Referral Whois Protocol (RWhois). Unter: ds.internic.net/ds/rfc1714.html, November 1994.

RFC 1738. T. Berners-Lee, L. Masinter, M. McCahill. Uniform Resource Locators (URL). Dezember 1994. Unter: www.faqs.org/rfcs/rfc1738.html

RFC 1808. R. Fielding. Relative Uniform Resource Locators. Juni 1995. Unter: www.faqs.org/rfcs/rfc1808.html

RFC 1835. Deutsch, P., Faltstrom, P., Schoultz, R. Weider, C. Architecture of the WHOIS++ service. Unter: Unter: www.faqs.org/rfcs/rfc1835.html, August 1995.

RFC 1866. T. Berners-Lee, D. Connolly. Hypertext Markup Language - 2.0. IETF Secretariat - Corporation for National Research Initiatives, März 1995. Unter: ds.internic.net/ds/rfc1866.html

RFC 1913. Fullton, J., Spero, S., Weider, C. Architecture of the Whois++ Index Service. Unter: ds.internic.net/ds/rfc1913.html, Februar 1996.

RFC 1945. T. Berners-Lee, R. Fielding, H. Frystyk. Hypertext Transfer Protocol -- HTTP/1.0. Mai 1996. Unter: www.faqs.org/rfcs/rfc1945.html

RFC 2068. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee. Hypertext Transfer Protocol -- HTTP/1.1. Januar 1997. Unter: www.faqs.org/rfcs/rfc2068.html

Internet-Drafts

Internet-Draft-Index. Current Internet-Drafts. IETF Secretariat - Corporation for National Research Initiatives. Unter: www.ietf.cnri.reston.va.us/1id-abstracts.html

Adler, Berglund et.al. (2000). Adler, S. Berglund, A., Caruso, J., Deach, S., Grosso, P., Gutentag, E., Milowski, A., Parnell, S., Richman, J., Zilles, S. Extensible Stylesheet Language (XSL), Version 1.0, W3C Candidate Recommendation November 2000. Unter: www.w3.org/TR/xsl/

Berners-Lee (1993). Berners-Lee, Tim. Hypertext transfer protocol. Unter: info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html

Berners-Lee (1995). Tim Berners-Lee. HyperText Markup Language Specification - Version 3.0. W3 Consortium, 1995. Unter: www.w3.org/pub/WWW/MarkUp/html3/CoverPage.html

Berners-Lee, Connolly et.al. (1995). Tim Berners-Lee, Dan Conolly, Wayne Gramlich, Jonathan Hirschman, Charlie Kindel, Lou Montulli, Eric Sink. Inserting multimedia objects into HTML3. W3C Working Draft, Dezember 1995.

Unter: www.w3.org/pub/WWW/TR/WD-insert-951221.html
Bos, Jacobs et.al. (1998). Bos, B., Jacobs, I., Lie, H., Lilley, C. Cascading Style Sheets, level 2, CSS2 Specification. W3C Recommendation Mai 1998. Unter: www.w3.org/TR/REC-CSS2
HTTP (1996). T. Berners-Lee, R. Fielding, H. Nielsen. Hypertext Transfer Protocol -- HTTP/1.1. Internet-draft, Januar 1996.
Unter: ftp://ietf.cnri.reston.va.us/internet-drafts/draft-ietf-http-v11-spec-01.txt
Burchard, Raggett (1995). P. Burchard, D. Raggett. Compound Documents in HTML. Internet-draft, November 1995.
Unter: ftp://ietf.cnri.reston.va.us/internet-drafts/draft-ietf-html-cda-00.txt
Maloney, Quin (1996). Maloney, M., Quin, L. Hypertext links in HTML. Internet-draft, Januar 1996. Unter: ftp://ietf.cnri.reston.va.us/internet-drafts/draft-ietf-html-relrev-00.txt
Raggett (1996). D. Raggett. HTML Tables. Internet-draft, Januar 1996.
Unter: ftp://ietf.cnri.reston.va.us/internet-drafts/draft-ietf-html-tables-06.txt
Ragett (1997). Ragett, Dave. HTML 3.2 Reference Specification. Unter:
www.w3.org/pub/WWW/TR/REC-html32.html
SHTTP (1996). E. Rescorla, A. Schiffman. The Secure HyperText Transfer Protocol. Internet-draft, Terisa Systems, Inc., Mai 1996 (Expires November-96).
Unter: ftp://cnri.reston.va.us/internet-drafts/draft-ietf-wts-shttp-02.txt
SSL (1996). Alan O. Freier, Philip Karlton, Paul C. Kocher. The SSL Protocol - Version 3.0. März 1996. Unter: ftp://ietf.cnri.reston.va.us/internet-drafts/draft-freier-ssl-version3-01.txt
X.509. S. Farrell, W. Ford, R. Housley, D. Solo. X509: Internet Public Key Infrastructure. November 1995. Unter: ftp://ietf.cnri.reston.va.us/internet-drafts/draft-ietf-pkix-ipki-00.txt

International Organisation for Standardisation (ISO)

ISO-Index. International Standards. International Organisation for Standardisation (ISO), März 1996. Unter: www.iso.ch/cate/cat.html
ISO 646. Information technology - ISO 7-bit coded character set for information interchange. Genf, ISO, 1991.
ISO 704. Principles and methods of terminology. Genf, ISO, 1987.
ISO 1087. Terminology - Vocabulary. Genf, ISO, 1990.
ISO 1087-2 (Committee Draft). Terminology work - Vocabulary - Part 2: Computational aids in terminology. Genf, ISO/TC 37/SC 3/N 127, 1993.
ISO 1951. Lexicographical symbols particularly for use in classified defining vocabularies. Genf, ISO, 1973.
ISO 2709. Documentation - Format for bibliographic information interchange on magnetic tape. Genf, ISO, 1981.
ISO 2788. Documentation - Guidelines for the establishment and development of monolingual thesauri. Genf, ISO, 1986.
ISO 5127-2. Documentation and information - Vocabulary - Part 2 : Traditional documents. Genf, ISO, 1983.
ISO 5127-3A. Documentation and information - Vocabulary - Section 3a) : Acquisition, identification, and analysis of documents and data. Genf, ISO, 1981.
ISO 5127-6. Documentation and information - Vocabulary - Part 6 : Documentary languages. Genf, ISO, 1983.
ISO 5127-11. Documentation and information - Vocabulary - Part 11 : Audio-visual documents. Genf, ISO, 1987.
ISO 5963. Documentation - Methods for examining documents, determining their subjects, and selecting indexing terms. Genf, ISO, 1985.

ISO 5964. Documentation - Guidelines for the establishment and development of multilingual thesauri. Genf, ISO, 1985.
ISO 6156. Magnetic tape exchange format for terminological/lexicographical records (MATER). Genf, ISO, 1987 (E).
ISO 8777. Documentation - Commans for interactive text searching. Genf, ISO, 1993.
ISO 8859. Information technology - 8-bit single-byte coded graphic character sets (Part 1-10). Genf, ISO, 1987-1992.
ISO 8879. Information Processing - Text and office systems - Standard Generalized Mark-up Language (SGML). Genf, ISO, 1986.
ISO 9075. Database Language SQL. - Part 3: Call-Level Interface (SQL/CLI), Part 4: Persistent Stored Modules (SQL/PSM). Genf, ISO, 1992 - 1996.
ISO 9594. X.500 The Directory; Part 1: Overview of Concepts; Part 2: X501: Models; Part 3: X.511 Abstract Service; Part 4: X.518 Procedures for Distributed Operation; Part 5: X.519 Protocol Specifications; Part 6: X.520 Selected Attribute Types; Part 7: X.521 Selected Object Classes; Part 8: X.509 Authentication Framework; Part 9: X.525 Replication; Part 10: Use of Systems Management for Administration of the Directory. Genf, ISO, 1995.
ISO 10021. MOTIS: Message oriented text interchange system. Genf, 19??
ISO 10162. Information and documentation - Open Systems Interconnection - Search and Retrieve Application Service Definition. Genf, ISO, 1993. siehe auch ANSI Z39.50.
ISO 10163-1.Information and documentation - Open Systems Interconnection - Search and Retrieve Application Protocol Specification - Part 1: Protocol specification Genf, ISO, 1993. siehe auch ANSI Z39.50.
ISO/DIS 10163-2. Information and documentation - Open Systems Interconnection - Search and Retrieve Application Protocol Specification - Part 2: Protocol Implementation Conformance Statement (PICS) proforma. siehe auch ANSI Z39.50.
ISO 10646. Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane. Genf, ISO, 1993.
ISO 10744. Information Technology - Hypermedia/Time-based Structuring Language (HyTime). Genf, ISO, 1992. oder unter: ftp://imgftp.uml.edu/pub/hytime/ oder unter: ftp://ftp.ifi.uio.no./pub/SGML/HyTime/HyQ-1.1.Kimber
ISO 12200 (Committee Draft). Computational aids in terminology - Terminological interchange format (TIF) - An SGML application. Genf, ISO/TC 37/SC 3/WG 3/N9, 1993.
ISO 12620 (Working Draft). Computational aids in terminology - Terminology interchange format (TIF) - Data element dictionary. Genf, ISO/TC 37/SC 3/WG 1/N 7, 1993.
ISO 13522-1. Information technology -- Coding of Multimedia and Hypermedia information (Part 1): MHEG object representation - Base notation. Genf, ISO/IEC DIS 1996.
ISO 13522-3. Information technology -- Coding of multimedia and hypermedia information (Part 3): MHEG script interchange representation. Genf, ISO/IEC DIS 1996.
ISO 13522-4. Information technology -- Coding of multimedia and hypermedia information (Part 4): Registration procedure for MHEG format identifier. Genf, ISO/IEC DIS 1996.
ISO 13522-5. Information technology -- Coding of multimedia and hypermedia information (Part 5): Support for Base-Level Interactive Applications. Genf, ISO/IEC DIS 1996.
ISO SQL 3. Database Language SQL/Foundation (SQL3) Part 1- 5, August 1994 (X3H2 Working Draft). unter: ftp.digital.com:80/pub/standards/sql/

Internationale Arbeitsgruppen für Standardisierung

Internet Standardization Group: HTML
Joint Technical Committee ISO/IEC JTC 1: Information Technology

Technical Committee ISO TC 37: Terminology
Technical Committee ISO TC 46: Documentation

10.4 Symbole

Mengen

$M = \{X: E(X)\}$	Menge aller Objekte X mit der Eigenschaft E
$X \in M$	x ist Element der Menge M
$M^1 \subseteq M^2$	M_1 ist (echte) Teilmenge von M_2
$M^1 \cup M^2$	Vereinigungsmenge von M_1 und M_2
$M^1 \times M^2 \times \ldots \times M^n$	n-faches kartesisches Produkt
$\langle M_1, M_2, \ldots, M_n \rangle$	Tupel aus $M_1 \times M_2 \times \ldots \times M_n$
$R \subseteq M^1 \times M^2 \times \ldots \times M^n$	n-stellige Relation über $M_1 \times M_2 \times \ldots \times M_n$

Logik

$\neg$	nicht
$\wedge$	und
$\vee$	oder
$\rightarrow$	Implikation
$\leftrightarrow$	Äquivalenz
$\exists$	es gibt ein
$\forall$	für alle
λ	alle, die (Lambda-Operator)
$\Im$	Interpretation
$C^{s1,\ldots sn}$	Prädikatenkonstante der Sorten $s_1, \ldots, s_n$
$P^{s1,\ldots sn}$	Prädikatenvariable der Sorten $s_1, \ldots, s_n$
$k^{s1,\ldots sn:s}$	Funktionskonstante von Sorten $s_1, \ldots, s_n$ auf die Sorte s
$f^{s1,\ldots sn:s}$	Funktionsvariable von Sorten $s_1, \ldots, s_n$ auf die Sorte s

10.5 URL-Echo-Sounder

10.5.1 Einführung und Grundlagen

URL-Echo-Sounder (URLES) ist nicht performant. Er dient lediglich dazu, die prinzipielle Machbarkeit zu zeigen. URLES ist ein Prototyp eines Internet-Informationssystems, das es dem Benutzer gestattet, Beziehungen zwischen Dokumenten ("connections") und Nachfahren von Dokumenten ("descendants") zu betrachten. Die benutzte Anfragesprache URLESQL ist eine kleine Teilmenge der im vorigen Kapitel beschriebenen universellen Anfragesprache.
URLES besteht aus einem Client (Applet) und einem Server. Der Client erlaubt dem Benutzer seine nach URLESQL spezifizierte logische Anfrage zu stellen. Der Server beantwortet die Anfrage und schickt dem Client das Ergebnis HTML-codiert zurück.

Java
Für die Realisierung von URL Echo Sounder wurde die Programmiersprache Java aus den folgenden Gründen ausgewählt (vgl. Middendorf, Singer, Strobel (1996)):
1. Java ist verteilt. Es lassen sich verteilte Client-Server-Anwendungen im Internet erstellen. Java stellt Methoden zur Kommunikation über TCP/IP und über WWW-Protokolle zur Verfügung.
2. Java ist portabel. Java-Anwendungen sind auf Serverseite mit Hilfe sogenannter virtueller Maschinen, die für alle häufig benutzten Hardwareplattformen als Interpreter verfügbar sind, hardwareunabhängig. Auf Clientseite wird die Hardwareunabhängigkeit durch die Einbindung des Java-Interpreters in Internet-Browsern erreicht, die wiederum für alle häufig benutzten Hardwareplattformen verfügbar sind.
3. *"Java ist vollständig objektorientiert aufgebaut. Es ähnelt in weiten Zügen C++, verzichtet jedoch auf deren prozeduralen Hintergrund. Nur so läßt sich ein sicheres Konzept realisieren." (Middendorf, Singer, Strobel (1996) S. 6)*
Der Umfang der Klassenbibliotheken ist zwar noch nicht so groß wie bei Smalltalk, es wird jedoch eine starke Vergrößerung in kurzer Zeit erwartet. Die erste verfügbare Klassenbibliothek von Java (JDK 1.0) hatte einen Umfang von 211 Klassen. Die neue Klassenbibliothek (JDK 1.1) hat einen Umfang von 503 Klassen.
4. Java ist sicher. *"Der Java-Interpreter im Inter-Browser enthält einen Verifizierer, der das Programm vor dem Ablauf nach einigen Regeln hin überprüft. Zur Authentifizierung werden kryptographische Verfahren eingesetzt, mit denen die Herkunft und Zugehörigkeit einzelner Applets zu einem Hersteller überprüft werden können." (Middendorf, Singer, Strobel 1996, 9)*
Client-Anwendungen (Applets) können über das Netzwerk nicht auf das lokale Dateisystem zugreifen, keine lokalen Bibliotheken laden, keine lokalen Fenster ohne Warnung öffnen und auch nicht die lokale Umgebung abfragen. Weiterhin kann ein Applet eine Socketverbindung nur zu einem Rechner aufbauen, von dem es geladen wurde (Applet-host = Socket-host). Netzverbindungen zu anderen Rechnern sind nicht zugelassen (vgl. iX 5/1996 S. 73).
5. Java ist robust. Es verzichtet auf das Sprachelement Pointer, auf Speicherverwaltungsfunktionen wie malloc, free etc (stattdessen automatische Speicherfreigabe), auf das Überladen von Methoden, auf Mehrfachvererbung und auf automatische Typumwandlung. Gleichzeitig ist Java streng objektorientiert hinsichtlich der Typüberprüfung.
6. Java ist parallel. Java-Programme können in mehrere einzelne parallele Prozeße unterteilt werden. Dafür werden Methoden zur Interprozeßkommunikation angeboten.
7. Java läßt sich einfach mit Datenbanken koppeln, beispielsweise mit JDBC.
8. Komplexe Benutzeroberflächen sind einfach zu gestalten.

9. Vorhandene Software kann in Java als sogenannter native code eingebunden werden.

Gleichzeitig bestehen die folgenden Nachteile bei der Verwendung von Java:
1. Java ist keine logische Programmiersprache.
2. Java ist relativ jung. Für Java existieren zu wenige, zu kleine und nicht ausgereifte Programmbibliotheken. Java ist weiterhin relativ ineffizient. An einer Verbesserung wird momentan sowohl auf der Software- als auch auf der Hardwareseite gearbeitet.
3. Java läßt Netzwerkverbindungen bisher ausschließlich im Internet zu.

Für die Realisierung der Komponenten Client und Server wird Java-Sourcecode von Flanagan 1996 verwendet.

10.5.2 Connections und Descendants

Das zweistellige Prädikat "Conn" (wie connection) bezeichnet 2-stellige gerichtete Beziehungen zwischen Dokumenten. Das erste Argument bezeichnet das Dokument, von dem die Beziehung ausgeht, das zweite Argument bezeichnet das Dokument zu dem die Beziehung hinführt. Dokumente werden im World Wide Web mit Hilfe von URL lokalisiert, vorwärtsgerichtete Beziehungen werden jeweils ausgehend von einem Dokument mit Hilfe des HTML-Elements ANCHOR ("A") ausgedrückt. Rückwärtsgerichtete Beziehungen sind mit HTML aufgrund seiner starken Bindung an Dateisysteme nicht direkt möglich, Suchmaschinen wie Alta Vista (Ray, Ray, Seitzer 1997) oder das Informationssystem Hyper-G (Dalitz, Heyer (1995), Maurer (1996)) bieten jedoch auch die Rückverfolgung von Beziehungen an, so daß dort folgende Frage gestellt werden kann: Welche Dokumente haben eine Beziehung zum Dokument mit der URL http://www.test.com/ ?
Mit Hilfe von Variablen für Dokumente bzw. URL und logischen Operatoren ist es möglich, Teilmengen von Beziehungsmengen (Teilgraphen) auszudrücken. Die Frage: Welche Dokumente haben eine Beziehung zu Dokumenten, die eine Beziehung zum Dokument "http://www.test.com/" haben , wird ausgedrückt als:
(λ Y) Conn("http://www.test.com/" , X) ∧ Conn(X, Y)
Mit den bisherigen Anfragemöglichkeiten läßt sich jedoch die folgende Frage nicht formulieren: Welche Nachfahren hat das Dokument "http://www.test.com/" ? Wir erweitern unsere Sprache um das 3-stellige Prädikat "Desc" (wie descendants). Das erste Argument bezeichnet das Dokument, von der die Beziehungen zu den Nachfahren ausgeht, das zweite Argument das Dokument, die ein Nachfahre des ersten Dokuments ist und das dritte Argument die Tiefe, bis zu der die Nachfahren bestimmt sind. Die obige Frage würde dann ausgedrückt als: (λ X) Desc("http://www.test.com/" , X , s).
Das Ergebnis einer Anfrage ist eine Menge von Dokumenten (URL).

10.5.3 URL-Echo-Sounder-Query-Language (URLESQL)

Im folgenden definieren wir Syntax und Semantik der in URL-ECHO-Sounder verwendeten Anfragesprache. Im Anhang geben wir die Syntax der Sprache zusätzlich in Backus-Naur Form.
Anfragen können wir mit der Benutzeroberfläche (Client) von URL-Echo-Sounder eingegeben. Die Interpretation der Anfrage wird dann vom Server von URL-Echo-Sounder bestimmt und im Client dargestellt.

Syntax

Alphabet:
1. Individuenkonstanten für eindeutige Namen von einfachen Dokumenten (URL): url_1 , url_2 , url_3 , ...
2. Individuenvariablen für eindeutige Namen von einfachen Dokumenten (URL):
x, y, z
3. Individuenkonstanten für die Suchtiefe: 1, 2, 3, ...
4. Prädikatenkonstanten für Beziehungen zwischen Dokumenten (2-stellig) und für Nachfahren bis zu einer Stufe (3-stellig): Conn, Desc
5. Logische Symbole: $\wedge$ (logisches und), $\vee$ (logisches oder), λ (alle, die)
6. Technische Symbole: die 3 Symbole: (,)

Formeln:
1. Conn(url_1 , url_2) und Mischungen mit den Individuenvariablen (z.B.: Conn(url_1 , x)) sind Formeln.
2. Desc(url_1 , url_2 , 1) und Mischungen mit den Individuenvariablen und der Suchtiefe (z.B.: Desc(url_1 , y, 1)) sind Formeln.
3. Wenn F_1 und F_2 Formeln sind, dann auch $F_1 \wedge F_2$ und $F_1 \vee F_2$.
4. Das sind alle Formeln.

Anfragen:
1. Wenn F eine Formel ist, dann ist (λ x) F eine Anfrage.
2. Das sind alle Anfragen.

Semantik

URL sei eine nichtleere Menge von URL (nach RFC 1738).
NAT sei eine nichtleere Menge von natürlichen Zahlen für die Suchtiefe

Interpretation der nichtlogischen Symbole:
$\Im(X) \in$ URL, X ist Individuenvariable oder Individuenkonstante für URL
$\Im(N) \in$ NAT, N ist Individuenkonstante für die Suchtiefe
$\Im(Conn) \subseteq URL \times URL$
$\Im(Desc) \subseteq URL \times URL \times NAT$

Formeln:
$\Im$ ist Modell von Conn(X,Y) gdw $<\Im(X), \Im(Y)> \in \Im(Conn)$
$\Im$ ist Modell von Desc(X,Y, N) gdw $<\Im(X), \Im(Y), \Im(N)> \in \Im(Desc)$
$\Im$ ist Modell von $F_1 \wedge F_2$ gdw $\Im$ ist Modell von F1 und $\Im$ ist Modell von F_2.
$\Im$ ist Modell von $F_1 \vee F_2$ gdw $\Im$ ist Modell von F1 oder $\Im$ ist Modell von F_2.

Anfragen:
$\Im(\lambda X) F = \{URL: \Im^{URL}{}_X$ ist Modell von F$\}$.
wobei: X ist Individuenvariable für URL

10.5.4 Backus-Naur Form der Anfragesprache

=:= ist das Zeichen für Definition
| ist das Zeichen für Alternative
Wiederholungen von Ausdrücken werden mit [ ] umschlossen
elementare Zeichen werden mit ' ' umschlossen

Query	=:=	Lambda-Expr Pred-Expr
Lambda-Expr	=:=	'λ' Varlist
Varlist	=:=	Var | Var , [Var]
Var	=:=	'x' | 'y' | 'z'
PredExpr	=:=	Pred | Pred-Expr Pred-Opr Pred-Expr
Pred	=:=	Connection | Descendants
Connection	=:=	'Conn' '(' Var ',' Var ')' | 'Conn' '(' URL ',' Var ')' | 'Conn' '(' Var ',' URL ')' | 'Conn' '(' URL ',' URL ')'
Descendants	=:=	'Desc' '(' Var ',' Var ',' Int ')' | 'Desc' '(' URL ',' Var ',' Int ')' | 'Desc' '(' Var ',' URL ',' Int ')' | 'Desc' '(' URL ',' URL ',' Int ')'
URL	=:=	alle nach RFC 1738 bildbaren URL
Int	=:=	'1' | '2' | '3' | '4'
Pred-Opr	=:=	'∧' | '∨'

Die Backus-Naur Form für URL wird in Lucyga (1996) gegeben.

10.5.5 Architektur

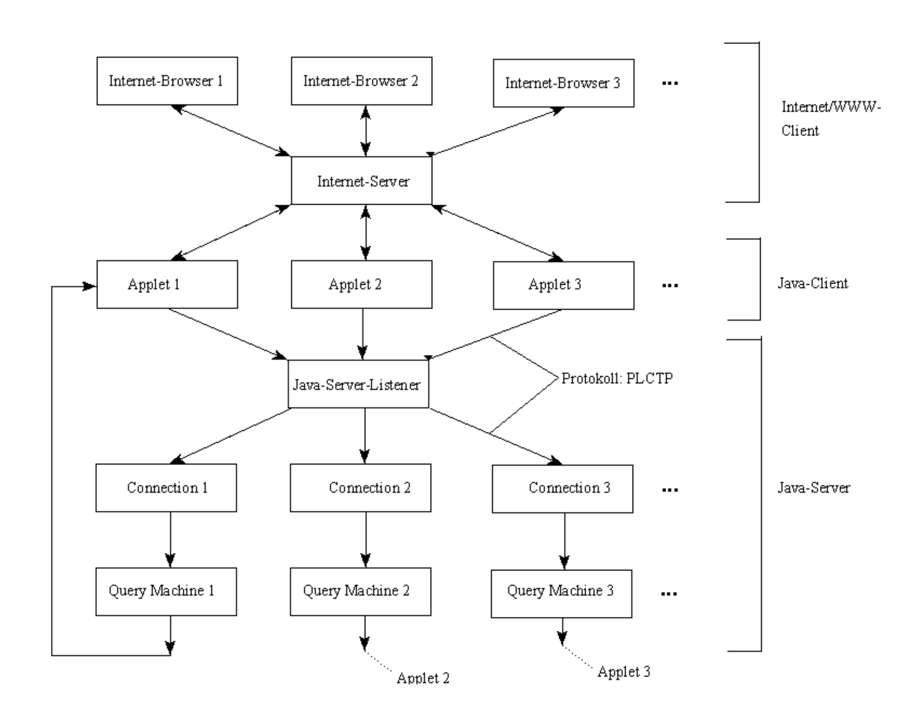


Abbildung 47: Architektur

10.5.6 Predicate Lambda Calculus Transfer Protocol (PLCTP)

PLCTP ist ein einfaches Sitzungsprotokoll (Schicht 5 des ISO/OSI-Modells) für Anfragen und Antworten, hier im Speziellen für prädikatenlogische Lambda-Anfragen. PLCTP ist zustandlos, da pro Sitzung genau eine Anfrage gestellt und genau eine Antwort bestimmt wird.
Unter der Sitzungsschicht liegen die Schichten 3 und 4 des Internet (TCP, IP) und die Verbindungs- und die Bitübertragungsschicht.
Die Kommunikation zwischen Server und Client wird durch in Java (Flanagan 1996, Middendorf, Singer, Strobel 1996) zur Verfügung gestellte Sockets realisiert. Unter den Sockets läuft

ein verläßliches und verbindungsbasiertes Kommunikationsprotokoll. Es wird eine semipermanente Verbindung zwischen den beiden Socketenden über einen Bytestrom aufgebaut.
Im Folgenden wird der Ablauf einer Sitzung mit PLCTP beschrieben:

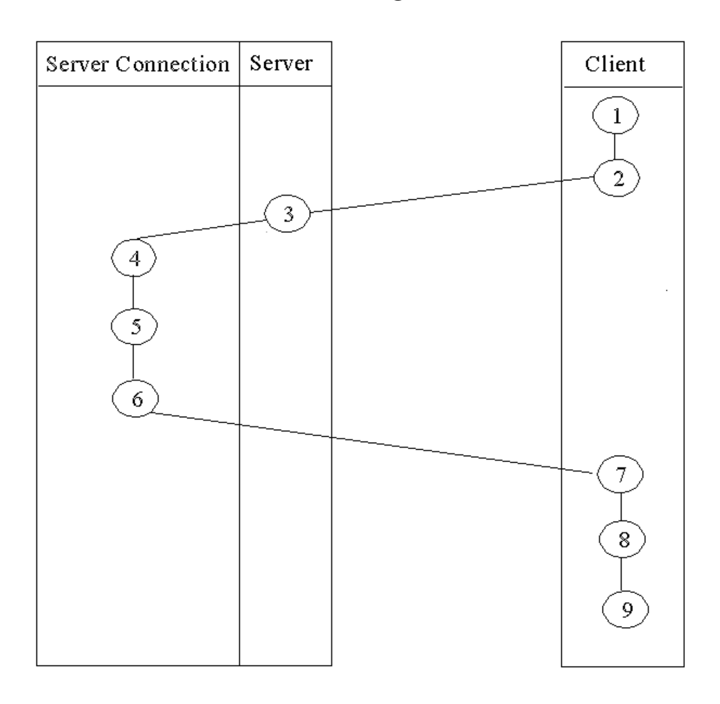


Abbildung 48: Eine PLCTP Sitzung

Die Kommunikation mit PLCTP verläuft in den folgenden Schritten:
1. Der Client wird initialisiert: Erzeugung des Client-Socket zum Server (unter Port 4712)
2. Der Client stellt eine Anfrage: über den Ausgangskanal der Socketverbindung wird die Anfrage an den Server geschickt.
3. Der Server erzeugt einen neuen Thread für die Anfrage, die Server-Connection.
4. Die Server Connection wird mit einem Server-Socket zum Client initialisiert.
5. Die Server-Connection liest die Anfrage über den Eingangskanal seiner Socketverbindung.
6. Die Server-Connection bestimmt die Antwort der Anfrage und schickt diese über den Ausgangskanal der Server-Socketverbindung an den Client.
7. Der Client liest die Antwort über den Eingangskanal seiner Socketverbindung.
8. Die Client- und Server-Sockets werden geschlossen. Die Server-Connection wird beendet.
9. Der Client stellt die Antwort dar.

Anfragen in URLESQL-Syntax werden vom Client bei der Übertragung zum Server aus Effizienzgründen in eine Infix-Notation überführt. Dabei werden technische Symbole weggelassen, logische Symbole in eine 7 Bit Zeichenfolge überführt (z.B. wird das Zeichen "∧" in die Zeichenfolge "and" überführt) und Lambda Variablen an das Ende des Ausdrucks gesetzt.
Beispielsweise wird die Anfrage:
(λ x y) Conn("http://www.test.com/" , x) ∧ Conn(x , y)
überführt in die Form:

```
and conn http://www.cs.tu-berlin.de/~josefw/ x conn x y x y
```

Antworten werden in der Hypertextbeschreibungssprache HTML in Form von geordneten Listen ("ul") codiert.

PLCTP kann in dem Maße als sicher bezeichnet werden, als daß Authentifizierung und Verschlüsselung auf Schicht 3 und 4 des OSI-Modells (sichere Socketverbindung) durch Java in naher Zukunft zur Verfügung gestellt werden.

10.5.7 Server

URLES-Server (URL Echo Sounder Server) ist ein Programm, das Verbindungen (Anfragen) von URLES-Clients annimmt, die Antwort bestimmt und an den Client zurückgibt.
URLES-Server hat grundsätzlich die folgenden Eigenschaften:
1. Er ist zuverlässig. Das Programm garantiert, daß Daten zuverlässig übertragen werden. Der Client sendet eine Anfrage an den Server und ist solange blockiert, bis der Server die Antwort sendet. Die Antwort ist quasi die Bestätigung der erfolgreichen Datenübertragung (vgl. Tanenbaum 1995 509).
2. Er blockiert nicht. Alle Anfragen werden an eigenständige Anfrage-Antwort-Threads (Klasse Connection) weitergereicht. URLES-Server kann dadurch mehrere Anfragen gleichzeitig bearbeiten. Clients brauchen nicht auf die Beendigung vorheriger Anfragebearbeitungen warten.
3. Er ist gepuffert. Die Kommunikation zwischen Server und Client wird mit einer Socketverbindung realisiert, deren Eingangs- und Ausgangskanal durch gepufferte Byteströme (nicht Paketströme) gebildet werden. Das Internet-Protokoll TCP/IP garantiert als Basis der Socketverbindung Zuverlässigkeit. Anfragen und Antworten können eine Folge von Bytes beliebiger Länge enthalten. Der Client sendet (momentan noch) 8-Bit-Zeichen nach ISO 8859-X an den Server, so daß der Vorteil der prinzipiellen 16-Bit-Fähigkeit von Java (Klasse Character) leider noch verlorengeht.
4. Er benutzt Internetadressen. Der Client lokalisiert den Server durch einen globalen Internet-Rechnernamen und eine eindeutige Nummer für den Serverprozeß auf diesem Rechner (port).
5. Er ist offen in Bezug auf Sicherheit. Authentifizierung und Verschlüsselung kann auf der Ebene der sicheren Socketverbindung einfach eingesetzt werden, da die Basisklassen in Java JDK 1.1 mittlerweile diese Funktionalität enthalten.

URLES-Server besteht aus den folgenden Klassen:
URLES-Server, Connection, Responder, Predicate, Variable, VarList, Link und HTMLList.URLES-Server und Connection sind Klassen, die die Kommunikation zwischen Client und Server realisieren. Responder, Predicate, Variable und VarList sind als Klassen für die Bearbeitung der logischen Anfrage zuständig:

a) Parsiere die Anfrage mit Hilfe der Klasse StringTokenizer.
b) Bestimme Prädikate, Variablen und logische Operatoren.
c) Bearbeite den logischen Anfrageausdruck (bestimme die Semantik); binde Variablen.

Link und HTMLList sind als Klassen für die Repräsentation von Hypertextstrukturen im Internet zuständig (Links und Listen).

Nachdem der Responder eine Anfrage erfolgreich bearbeitet hat, sendet er die Antwort HTML-codiert zum Client. Die Verbindung zwischen Server und Client wird entweder geschlossen oder es wird einen weitere Verbindung für eine neue Anfrage aufgebaut.

10.5.8 Client

URLES-Client (URL Echo Sounder Client) ist eine Benutzeroberfläche (Java-Applet), die es dem Benutzer erleichtert, eine Anfrage als Ausdruck des Prädikatenkalküls mit Lambda Variablen zu formulieren und die Antwort zu betrachten.

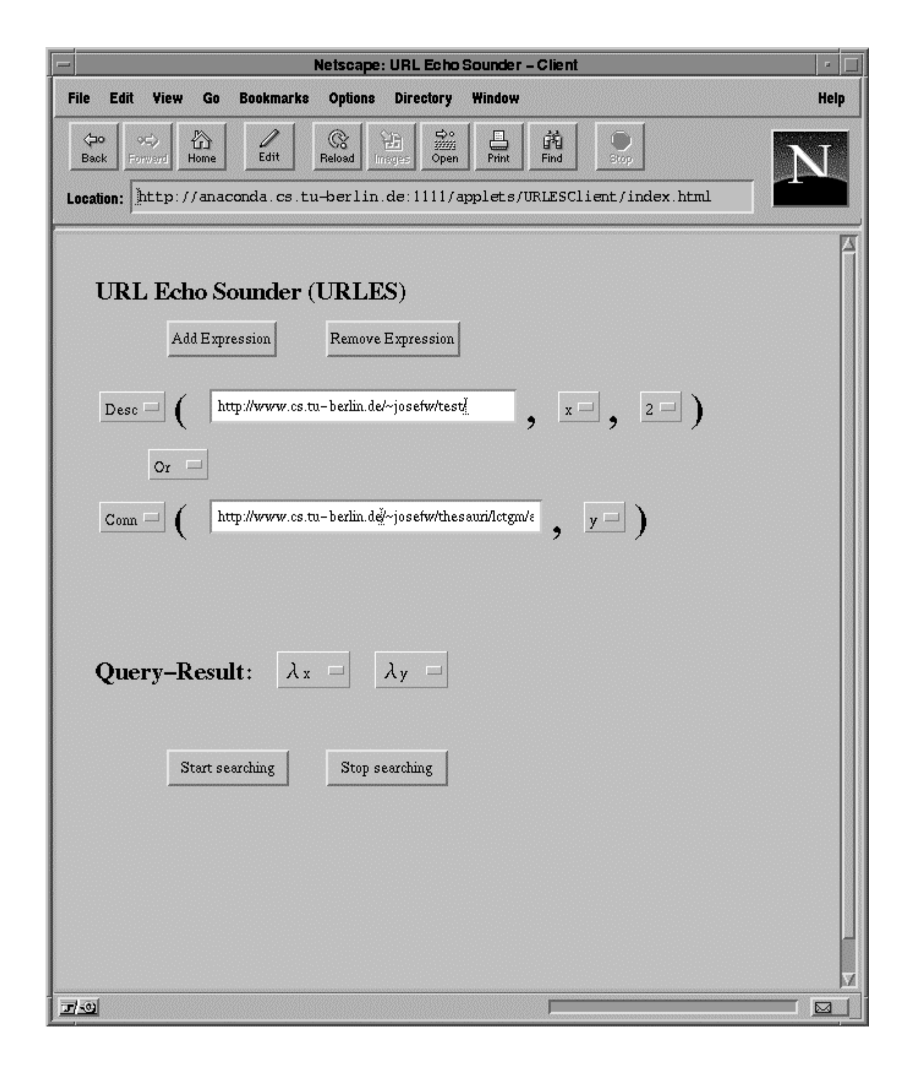


Abbildung 49: Formulierung einer Anfrage mit URLES (im März 1997)

Nachdem der Benutzer mit Hilfe eines Internet-Browsers das URLES-Java-Applet über einen WWW-Server geladen hat, spezifiziert er eine logische Anfrage weitgehend mit der Maus. Syntaktisch fehlerhafte Eingaben werden in speziellen Fenstern angezeigt. Danach startet der Benutzer seine Anfrage mit dem "Start-Searching" Knopf. Aus Sicherheitsgründen dürfen Applets Socketverbindungen nur zu einem Prozeß aufbauen, die auf dem gleichen Rechner (Server-host) laufen, wie der Rechner, von dem das Applet geladen wurde (Applet-host). Das Applet baut in einer Initialisierungsphase (Methode "initSocket") eine TCP/IP-basierte Socketverbindung zum URLES-Server (hier über Port 4712) auf. Weiterhin erzeugt das Applet einen Prozeß (Klasse "StreamListener"), das auf eine Antwort von URLES-Server über den Eingangskanal der Socketverbindung wartet. Für die Anfrage wird auf dem Server ein eigenständiger Anfrage-Thread mit dieser Socketverbindung aufgebaut. Das Applet übersetzt die graphische Anfrage in die vom Server geforderte Prefix-Notation. Diese wird dann über die Socketverbindung zum URLES-Server gesendet. Nachdem der URLES-Server die Anfrage bearbeitet hat, schickt er die Antwort über den Eingabekanal der Socketverbindung an den wartenden Prozeß im Applet zurück. Dieser empfängt die Antwort und stellt das Ergebnis mittels der z.B. mit dem Internet-Browser Netscape zur Verfügung gestellten Javascript Methoden in seinem Hauptfenster dar.

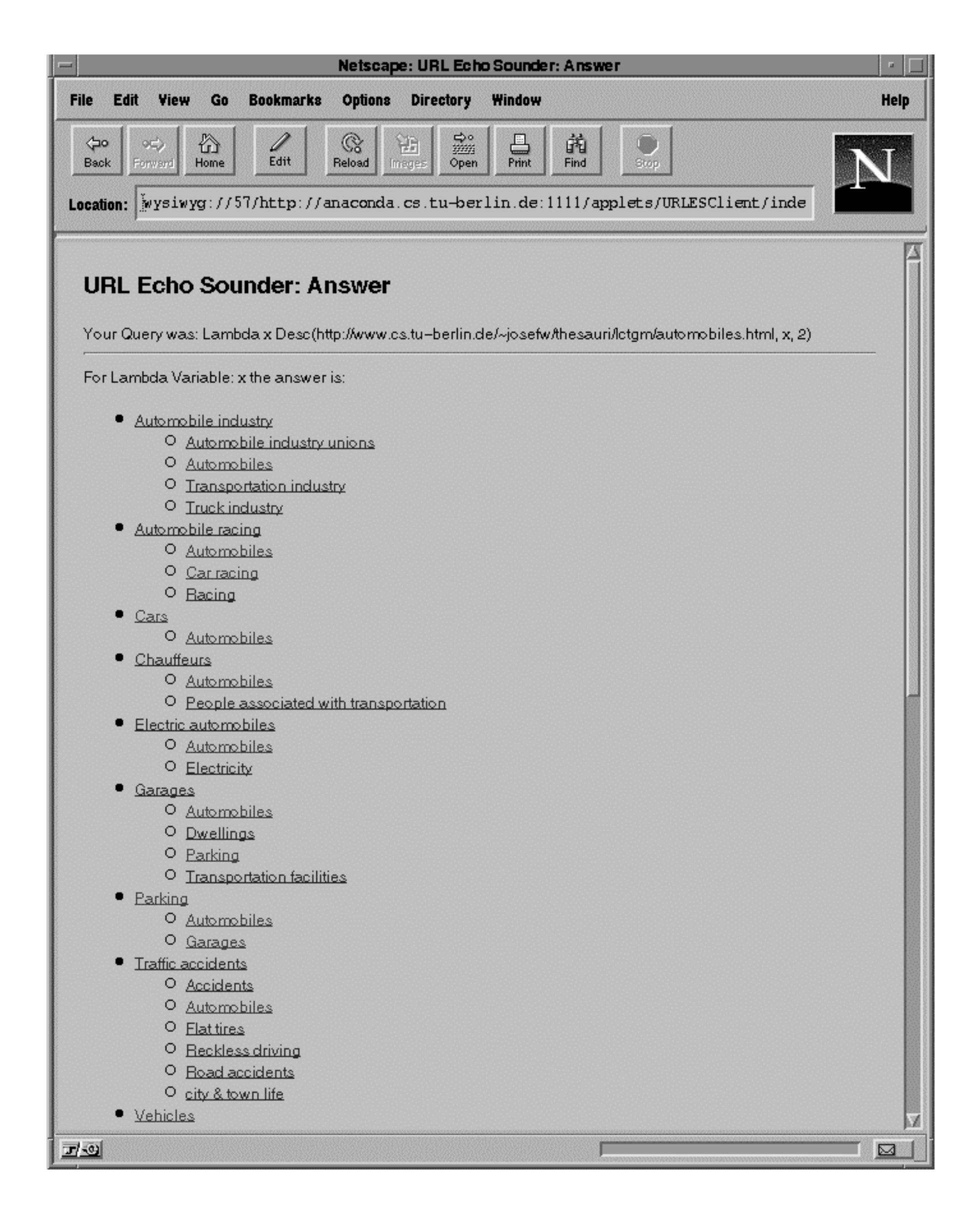


Abbildung 50: URLES-Anfrageergebnis als Hierarchie (im März 1997)

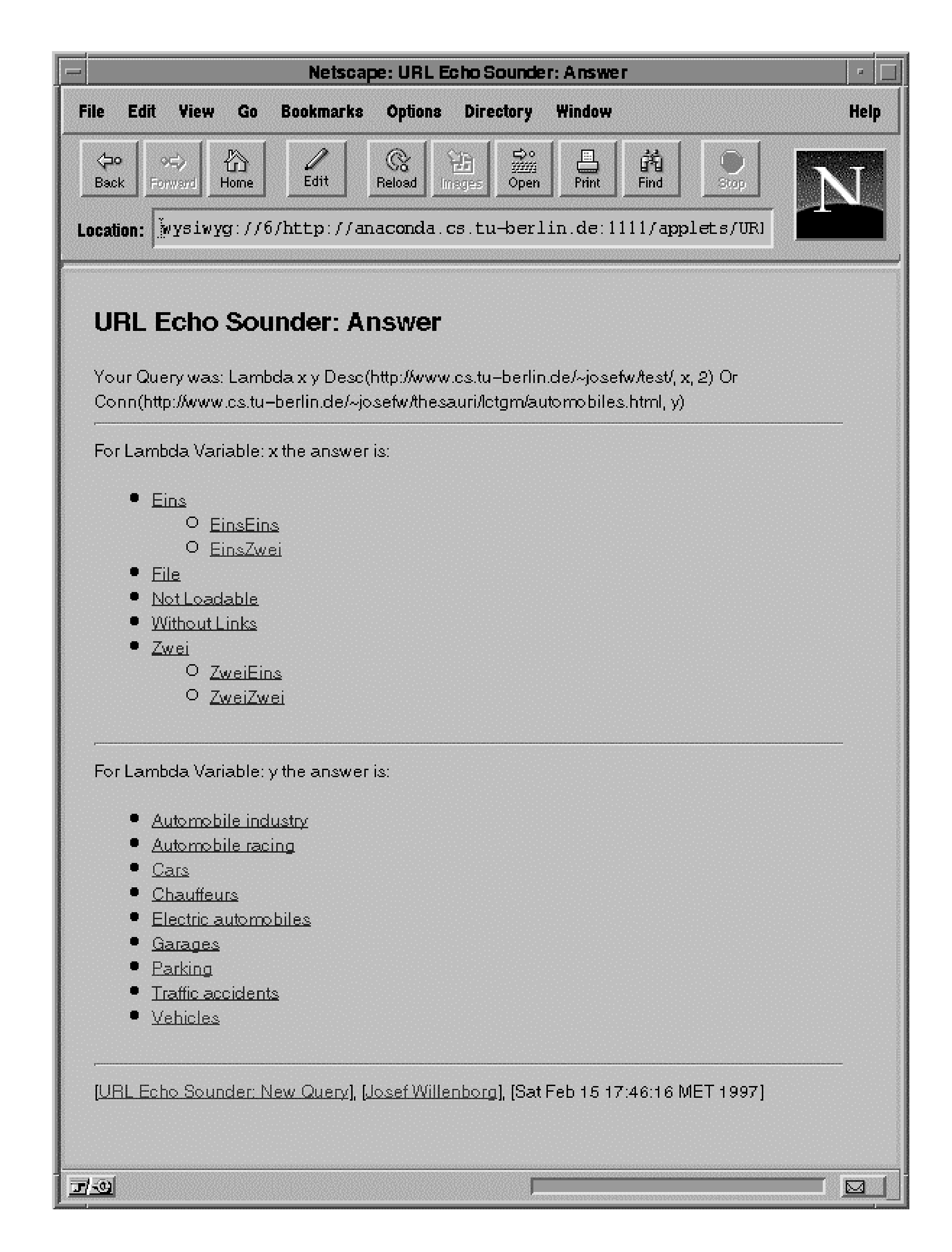


Abbildung 51: URLES-Anfrageergebnis mit zwei Variablen (im März 1997)

10.5.9 Erweiterungen

Momentan ist URLES begrenzt auf:
1. Zwei Prädikate ("Conn" und "Desc")
2. Suchtiefe 4 bei dem Prädikat "Desc"
3. ein Maximum von zwei logischen Ausdrücken
4. zwei logische Operatoren
5. drei Variablen
6. vorwärtsgerichtete Beziehungen

Zyklen werden zudem im Anfrageergebnis hierarchisch redundant dargestellt.

Punkt 3, 4 und 5 lassen sich relativ einfach erweitern, da in der Implementierung auf Erweiterbarkeit Wert gelegt wurde. Die Erweiterung von Punkt 1, 2 und 6 würde entweder die Ein

setzung einer zentralen Suchmaschine oder die dezentrale Realisierung mit Hilfe vorhandener Suchmaschinen verlangen.
Als Alternative zur Tiefensuche kann eine Breitensuche oder eine beschränkte Tiefensuche eingesetzt werden.

Ein Kompromiß zwischen einer Erweiterung der Mächtigkeit der Anfragemöglichkeiten, einer akzeptablen Effizienz und einer Einfachheit der Realisierung stellen die folgenden Vorschläge dar:

- Einbeziehung von Suchattributen (Datum, Protokoll)
- Einbeziehung von Volltextsuchfunktionen (z. B. Grep)
- Einbeziehung von Relationsarten in die Beziehungen zwischen Dokumenten
- Darstellung des Anfrageergebnisses als geordnete Menge oder als Hierarchie
- Ordnung des Anfrageergebnisses nach Kriterien: alphabetisch absteigend und aufsteigend, nach Zeitpunkt der Erstellung, nach dem Autor, nach der ursprünglichen Reihenfolge im Dokument etc.

Es könnte dann z.B. die folgende Anfrage gestellt werden:
Welche Dokumente sind Nachfolger des Dokuments http://www.thesaurus.com/auto.html bis zur Tiefe 5 bezüglich der Relations "NT" (narrower term), sind vom Protokolltyp "http", enthalten die Zeichenfolge "automobil" in ihrem Inhalt und haben einen Erstellungszeitraum zwischen 1995 und 1996 ? Das Ergebnis wird alphabetisch geordnet und erscheint als Hierarchie.
Als Lambda-Anfrage:

(λ x) Desc("http://www.thesaurus.com/auto.html" , x, z, 5)
∧ Relation("NT", z)
∧ Attribut("content", "automobil", x)
∧ Attribut("date", y, x)
∧ (Conn("1.1.1995", y, ">=") oder Conn(31.12.1996, y, "<="))
∧ Attribut("presentation", "alphabetically", x)
∧ Attribut("presentation", "hierarchy", x)

10.5.10 Programmlisting

Im Folgenden drücken wir jeweils durch ein „FLAN“ rechts neben die Zeile des Quellprogrammcodes aus, daß es sich um Code aus Flanagan (1996), Copyright © 1996 O’Reilly & Associates handelt. Für den eigenentwickelten Code gilt die GNU General Public License.

Die Datei: Server.java

```
import java.io.*;
import java.lang.*;
import java.net.*;
import java.util.*;

/* ----------- URL Echo Sounder (URLES) - Server --------------- */
/* URLES-Server is the main program that establishes a          */
/* connection (query) from a client to an original              */
/* query-response thread.                                       */
/* URLES-Server has the following properties:                   */
/* 1. it is reliable. The system guarantees that the data that   */
/*    is to be sent will be sent. The client sends a query       */
/*    and is blocked up till the server has sent the request.    */
/*    The answer of the server is quasi the confirmation of      */
/*    successful communication.                                 */
/* 2. it is not blocked up. All further requests can be executed */
/*    directly. For each query the main server process starts    */
/*    a new connection thread.                                  */
/* 3. it is buffered. Communication between server and client    */
/*    is established through a socket. Input and ouput of that   */
/*    is a buffered byte stream (not packet stream). TCP/IP      */
```

```
/*    as the basis for sockets guarantees their reliability.     */
/*    Queries and answers contain a sequence of bytes of any     */
/*    length. The client sends 8 Bit characters (ISO 8859-X) to  */
/*    the server. The 16 Bit availability of Java cannot be used.*/
/* 4. it uses internet addresses. The client is localizing the   */
/*    server with a global internet host name and an identifying */
/*    number for the server process (port) on that host.         */
/* 5. it is open to security. Authentification and cryptography  */
/*    are available for sockets with Java JDK 1.1.               */
/*                                                               */
/* URLESServer has the following classes:                        */
/* URLESServer, Connection, Responder, Predicate, Variable,      */
/* VarList, Link and HTMLList.                                   */
/* URLESServer and Connection are classes for communication      */
/* between client and server. Responder, Predicate, Variable,    */
/* and VarList are classes for executing the logical queries:    */
/* a) parse the query (with help of class StringTokenizer)       */
/* b) determine predicates, variables and logical operators      */
/* c) execute the logical expression (find semantics); bind      */
/*    variables                                                  */
/* Link and HTMLList are classes to represent the structure of   */
/* hypertext (links and lists).                                  */
/* After successful executing the logical query, the responder   */
/* sends the answer (HTML coded) to the client (through the      */
/* socket connection). The last step is to close the connection  */
/* to the client or to establish a new connection for the next   */
/* query.                                                        */
/*                                                               */
/* For further documentation look at the classes and methods.    */

public class Server extends Thread                                        FLAN
/* Main program loop to handle communication with the clients:   */
/* The class variable "DEFAULT PORT" names the port, if the user */
/* does not specify a port.                                      */
/* The instance variable "port" names the port-number and        */
/* "listen-socket" names the main server socket. If the client   */
/* knocks at the door (has a request), it creates a new          */
/* connection, which handles further communication with the      */
/* client. The server listens to further client requests. So it  */
/* is not blocked up.                                            */

{public final static int DEFAULT_PORT = 4712;                             FLAN
 protected int port;                                                      FLAN
 protected ServerSocket listen_socket;                                    FLAN

 public static void fail(Exception e, String msg)                         FLAN
 /* Exit with an error message, when an exception occurs.       *         FLAN
 {System.err.println(msg + ": " +  e);                                    FLAN
  System.exit(1);                                                         FLAN
 }                                                                        FLAN

 public Server(int port)                                                  FLAN
 /* Create a ServerSocket to listen for connections on;         */
 /* start the thread.                                           */
 {if (port == 0) {port = DEFAULT_PORT;};                                  FLAN
   this.port = port;                                                      FLAN
  try { listen_socket = new ServerSocket(port); }                         FLAN
  catch (IOException e) {fail(e, "Exception creating server socket");};   FLAN
  System.out.println("URLES-Server: listening on port " + port);          FLAN
  start();                                                                FLAN
 }                                                                        FLAN

 public void run()                                                        FLAN
 /* The body of the server thread. Loop forever, listening to    */
 /* and accepting connections from clients. For each connection, */
 /* creates a Connection object to handle communication through  */
 /* the new Socket.                                              */
 {try                                                                     FLAN
  {while(true)                                                            FLAN
    {Socket client_socket = listen_socket.accept();                       FLAN
     Connection c = new Connection(client_socket);                        FLAN
    }                                                                     FLAN
  }                                                                       FLAN
  catch (IOException e) {fail(e, "Exception while listening for connections");};
 }                                                                        FLAN

 public static void main(String[] args)
 /* Starts the server up and listens to an optionally specified  */
 /* port.                                                        */
 {int port = 0;
  if (args.length == 1)
   {try
     {port = Integer.parseInt(args[0]);}
    catch (NumberFormatException e) {port = 0;};
   }
  new Server(port);
 }
```

```
}

class Connection extends Thread                                       FLAN
/* This class is the thread that handles communication with one  */
/* client. The instance variable "clientSocket" names the         */
/* socket to the client, "in" names the buffered DataInputStream */
/* (byte stream) and "out" names the buffered PrintStream (byte  */
/* stream.                                                        */

{protected Socket clientSocket;                                       FLAN
 protected DataInputStream in;                                        FLAN
 protected PrintStream out;                                           FLAN

 public Connection(Socket s)                                          FLAN
 /* Initialize the streams and start the thread.                 */   FLAN
 {clientSocket = s;                                                   FLAN
  try                                                                 FLAN
   {in = new DataInputStream(clientSocket.getInputStream());          FLAN
    out = new PrintStream(clientSocket.getOutputStream());            FLAN
   }                                                                  FLAN
  catch (IOException e)                                               FLAN
   {try                                                               FLAN
     {clientSocket.close();}                                          FLAN
    catch (IOException e2)                                            FLAN
     {System.err.println("Exception while getting socket streams: " + e);}       FLAN
    return;                                                           FLAN
   }                                                                  FLAN
  start();                                                            FLAN
 }                                                                    FLAN

 public void run()                                                    FLAN
 /* Provides the service. Reads the URLES-query from the         */
 /* buffered input stream. Starts a responder to get the answer  */
 /* and sends it as HTML code back to the client through the     */
 /* buffered output stream.                                      */
 {String URLESQueryString, HTMLAnswerString;
  try                                                                 FLAN
   {for(;;)                                                           FLAN
     {/* reads a query-string, parses the line and sends the     */
      /* answer back to the client                               */
      URLESQueryString = in.readLine();
      Responder responder = new Responder(URLESQueryString);
      HTMLAnswerString = responder.getAnswer();
      out.println(HTMLAnswerString);
      clientSocket.close();                                           FLAN
     }                                                                FLAN
   }                                                                  FLAN
  catch (IOException e) {;}                                           FLAN
 }
}
class Responder extends Object
/* Initializes predicates, variables, lambda variables and       */
/* logical operator with the value of parsing the query. A query */
/* is a sequence (seperated by blanks) of elementary logical     */
/* query strings in prefix notation.                             */
{public VarList vars;
 public VarList lambdaVars;
 public Predicate pred1;
 public Predicate pred2;
 public String logOp;

 public Responder(String query)
 {StringTokenizer strTokenizer;
  String token;
  Predicate initPred = new Predicate("","","",1,"",this);
  pred1 = pred2 = initPred;
  vars = lambdaVars = new VarList();
  logOp = "";
  strTokenizer = new StringTokenizer(query);
  token = strTokenizer.nextToken();
  if (token.equals("Conn") || token.equals("Desc"))
    {strTokenizer = new StringTokenizer(query);
     initPred(strTokenizer, "first");}
  if (token.equals("Or") || token.equals("And"))
    {this.logOp = token;
     initPred(strTokenizer, "first");
     initPred(strTokenizer, "second");
    }
  initLambdaVars(strTokenizer);
  initVars();
 }
 public void initPred(StringTokenizer strTokenizer, String pos)
 /* parses the query for exactly one predicate. If the predicate */
 /* name is "Conn" it reads two token (for the two places of     */
 /* the predicate), if the predicate name is "Desc", it reads    */
 /* three token (for the three places of the predicate).         */
 {String predName, arg1, arg2, arg3, resultType;
```

```
 predName = arg1 = arg2 = arg3 = resultType = "";
 Predicate pred;
 int depth = 1;
 predName = strTokenizer.nextToken();
 if (predName.equals("Conn"))
  {arg1 = strTokenizer.nextToken();
   arg2 = strTokenizer.nextToken();
   resultType = "list";
  }
 if (predName.equals("Desc"))
  {arg1 = strTokenizer.nextToken();
   arg2 = strTokenizer.nextToken();
   arg3 = strTokenizer.nextToken();
   Integer depthInteger = new Integer(arg3);
   depth = depthInteger.intValue();
   resultType = "hierarchy";
  }
 pred = new Predicate(predName, arg1, arg2, depth, resultType, this);
  if (pos.equals("first"))
   {pred1 = pred;}
  if (pos.equals("second"))
   {pred2 = pred;}
}
public void initLambdaVars(StringTokenizer strTokenizer)
/* Parses the query for lambda variables. Generates an          */
/* alphabetically sorted list of lambda variables for the       */
/* instance variable "lambdaVars".                              */
{VarList result = new VarList();
 while (strTokenizer.hasMoreTokens())
  {Variable var = new Variable(strTokenizer.nextToken(), new HTMLList("ul"));
   result.addElement(var);}
 result.sort();
 lambdaVars = result;
}
public void initVars()
/* Generates an alphabetically sorted list of all variables in  */
/* all predicates for the instance variable "vars".             */
{VarList vars = new VarList();
 if (isVar(pred1.arg1))
  {Variable var = new Variable(pred1.arg1, new HTMLList("ul"));
   vars.addElement(var);}
 if (isVar(pred1.arg2))
  {Variable var = new Variable(pred1.arg2, new HTMLList("ul"));
   vars.addElement(var);}
 if (isVar(pred2.arg1))
  {Variable var = new Variable(pred2.arg1, new HTMLList("ul"));
   vars.addElement(var);}
 if (isVar(pred2.arg2))
  {Variable var = new Variable(pred2.arg2, new HTMLList("ul"));
   vars.addElement(var);}
 /* this dummy variable is needed for "logical and execution "  */
 Variable testVar = new Variable("g", new HTMLList("ul"));
 vars.addElement(testVar);
 vars.sort();
 this.vars = vars;
}
public boolean isVar(String v)
{if ("xyzg".indexOf(v) != -1)
   {return true;}
 else {return false;}
}
public String infixNotation()
/* Delivers an infix string from the query expression.          */
{String result = "";
 String lambdaVars = "";
 lambdaVars = lambdaVars();
 String pred1String = infixNotation(pred1);
 result = "Lambda " + lambdaVars + " " + pred1String;
 if (logOp.equals("And") || logOp.equals("Or"))
   {String pred2String = infixNotation(pred2);
    result = result + " " + logOp + " " + pred2String;
   }
 return result;
}
public String lambdaVars()
/* Delivers a string of all lambda variables seperated by       */
/* blanks.                                                      */
{int max = lambdaVars.size() - 1;
 StringBuffer buf = new StringBuffer();
 for (int i = 0; i <= max; i++)
  {Variable var = (Variable)lambdaVars.elementAt(i);
   buf.append(var.name);
   if (i < max)
     {buf.append(" ");}
  }
 return buf.toString();
}
```

```
 public String infixNotation(Predicate pred)
 /* Delivers an infix string of a predicate.                    */
 {String result = "";
  if (pred.isConn())
    {result = pred.name + "(" + pred.arg1 + ", " + pred.arg2 + ")";}
  if (pred.isDesc())
         {result  =  pred.name  +  "("  +  pred.arg1  +  ",  "  +  pred.arg2  +  ",  "  +
String.valueOf(pred.arg3) + ")";}
  return result;
 }
 public String getLambdaAnswer()
 /* Delivers an HTML coded answer string. Starts the execution of*/
 /* the logical query. Then variables are bound. The result is   */
 /* sent back to the client.                                     */
 {int max = lambdaVars.size() - 1;
  StringBuffer buf = new StringBuffer();
  for (int i = 0; i <= max; i++)
   {Variable var = (Variable)lambdaVars.elementAt(i);
    buf.append("For Lambda Variable: ");
    buf.append(var.name);
    buf.append(" the answer is:<br>");
    HTMLList lambdaHAnswer = (HTMLList)vars.getVar(var.name).value;
    String lambdaAnswer = lambdaHAnswer.toHTMLString();
    buf.append(lambdaAnswer);
    buf.append("<hr>");
   }
  return buf.toString();
 }
 public String getAnswer()
 {String date = new Date().toString();
        String    HTMLHeadString    =    "<HTML><HEAD><TITLE>URL    Echo    Sounder:    Ans-
wer</TITLE></HEAD><BODY><H1>URL Echo Sounder: Answer</H1>";
            String       HTMLEndString       =       "[<A       HREF='http://anaconda.cs.tu-
berlin.de:1111/applets/hyqlClient/index.html'>URL   Echo   Sounder:   New   Query</a>],   [<a
href='http://www.cs.tu-berlin.de/~josefw/'>Josef    Willenborg</a>],    ["    +    date    +
"]</BODY></HTML>";
  String infixQueryString = infixNotation();
  String HTMLQueryString = "Your Query was: " + infixQueryString + "<br><hr>";
  if (logOp.equals("Or"))
    {pred1.or(pred2);}
  if (logOp.equals("And"))
    {pred1.and(pred2);}
  if (logOp.equals(""))
    {pred1.execute();}
  String lambdaAnswer = getLambdaAnswer();
  String queryResult = HTMLHeadString + HTMLQueryString + lambdaAnswer + HTMLEndString;
  return queryResult;
 }
}

class HTMLList extends Vector
/* This class is a vector (ordered list of HTMLLists). It is     */
/* used to represent an HTML list (<ul>, <ol> etc.). It         */
/* recursively contains HTMLLists down to the leafs. In one      */
/* hierarchy level, the entries are ordered alphabetically,      */
/* double entries (if link.source and link.dest of two entries   */
/* are equal) are removed.                                       */
/* The instance variable "type" names the type ("ul", "ol", ...) */
/* of the HTMLList, "link" names the entry link and "depth" names*/
/* the depth of hierarchically deeper entries.                   */
/* An HTMLList is a directed, cyclic graph of links.             */
{public String type;
 public Link link;
 public int depth;

 public HTMLList(String type)
  {super();
   this.type = type;
  }
 public HTMLList(String type, Link link, int depth)
  {super();
   this.type = type;
   this.link = link;
   this.depth = depth;
  }
 public void setLink(Link link)
 {this.link = link;}
 public boolean isRoot()
 {if (link.source.equals("{}{}ROOT{}{}"))
    {return true;}
  else {return false;}
 }
 public void addObject(Object e)
 {/* adds an Object (HTMLList) alphabetically sorted into this   */
  /* HTMLList. Caution: removes double Entries (if link.source   */
  /* and link.dest of two entries are equal.                     */
  Link link1 = new Link("Error","Error");
```

```
 Link link2 = new Link("Error","Error");
 int max = size() - 1;
 /* If this HTMLList is empty, insert e.                        */
 if (max == -1)
   {addElement(e);
    return;
   }
 if ( e instanceof HTMLList )
   {HTMLList ul = (HTMLList)e;
    link1 = ul.link;}
 {for (int i = 0; i <= max; i++)
   {Object elem = elementAt(i);
    if ( elem instanceof HTMLList )
     {HTMLList ul = (HTMLList)elem;
      link2 = ul.link;}
    /* if elem1 is equal to elem2, remove it from HTMLList      */
    if (link1.source.compareTo(link2.source) == 0 &&
                  link1.dest.compareTo(link2.dest) == 0 )
       {return;}
    /* if link1.source is lexically smaller than link2.source   */
    if (link1.source.compareTo(link2.source) < 0)
       {insertElementAt(e, i);
        return;}
    /* if link1 is lexically is the last element                */
    if (i == max)
       {addElement(e);
        return;}
   }
 }
}
public HTMLList union(HTMLList l)
/* Generates a HTMLList which unions this HTMLList with l. The  */
/* result is sorted alphabetically and does not contain double  */
/* entries.                                                     */
{int max = l.size() - 1;
 for (int i = 0; i <= max; i++)
   {addObject(l.elementAt(i));}
 return this;
}
public void diff(HTMLList l)
/* subtract all elements in l from this HTMLList. If an element */
/* from l is not a member in this HTMLList, doesnt`t do         */
/* anything.                                                    */
{int max = l.size() - 1;
 for (int i = 0; i <= max; i++)
   {removeElement(l.elementAt(i));}
}
public String toHTMLString()
/* recursively generates a HTML string from this HTMLList       */
{if (size() == 0)
   {return "";}
 StringBuffer buf = new StringBuffer();
 if (isRoot())
  {buf.append("\n" + "<" + type + ">" + "\n");}
 else {buf.append("<li>" + link.toHTMLString() + "\n" + "<" + type + ">" + "\n");}

 int max = size() - 1;
 Enumeration elements = elements();
 for (int i = 0 ; i <= max ; i++)
  {Object elem = elements.nextElement();
   HTMLList ul = (HTMLList) elem;
   if (ul.size() == 0)
    {buf.append("<li>" + ul.link.toHTMLString() + "\n");}
   if (ul.size() > 0)
     {buf.append(ul.toHTMLString() + "\n");}
  }
 buf.append("</" + type + ">" + "\n");
 return buf.toString();
 }
public HTMLList toSet()
/* generates a set from a hierarchy of this HTMLList            */
{HTMLList result = new HTMLList("ul", link, 1);

 int max = size() - 1;
 Enumeration elements = elements();
 for (int i = 0 ; i <= max ; i++)
  {Object elem = elements.nextElement();
   HTMLList ul = (HTMLList) elem;
   if (ul.size() == 0)
    /* to add the leaf nodes                                    */
    {result.addObject(ul);}
   if (ul.size() > 0)
     {result.union(ul.toSet());
      /* to add as well the other nodes (not-leaf nodes)        */
      ul.removeAllElements();
      result.addObject(ul);
     }
```

```
  }
 return result;
 }
public String getDocumentString(URL url)                               FLAN
{/* Loads document for URL url and returns this document as a   */
 /* string.                                                      */
 String documentString, line;                                          FLAN
 String result = "";                                                   FLAN
 documentString = "";                                                  FLAN
 URLConnection conn;                                                   FLAN
 DataInputStream in;                                                   FLAN
 InputStream connIn;                                                   FLAN
 try                                                                   FLAN
  {conn = url.openConnection();                                        FLAN
   conn.connect();                                                     FLAN
   connIn = conn.getInputStream();                                     FLAN
   in = new DataInputStream(connIn);                                   FLAN
   while ((line = in.readLine()) != null)                              FLAN
    {documentString = documentString + line + "\n";}                   FLAN
   in.close();                                                         FLAN
   return documentString;                                              FLAN
  }                                                                    FLAN
 catch (IOException e2) {return result;}                               FLAN
}                                                                      FLAN
public Link getLink(URL sourceURL, String destURLString)
{/* Delivers a link from sourceURL to destURLString. "source"   */
 /* "dest" of link are corrected (blanks, quotations, ...) and  */
 /* incomplete URL strings are filled (relative href, no "/" at */
 /* end, ...                                                     */
 Character quote1Char = new Character('"');
 char quote1 = quote1Char.charValue();
 Character quote2Char = new Character(''');
 char quote2 = quote2Char.charValue();
 Character equalChar = new Character('=');
 char equal = equalChar.charValue();
 Character greaterChar = new Character('>');
 char greater = greaterChar.charValue();
 String s, urlToString, lastString, newURLString;
 int start, end, sEnd, length;

 String hrefString = "";

 /* "=" and ">" are limits of the URL string                     */
 start = destURLString.indexOf(equal);
 end = destURLString.indexOf(greater, start+2);
 /* s is the hrefString with quotation marks.                    */
 s = destURLString.substring(start+1, end).trim();
 sEnd = s.length();
 /* 4 tests, if URL has quotation marks around.                  */
 if ((s.charAt(0) == quote1) && (s.charAt(sEnd-1) == quote1))
  {hrefString = s.substring(1,sEnd-1);}
 if ((s.charAt(0) != quote1) && (s.charAt(sEnd-1) == quote1))
  {hrefString = s.substring(0,sEnd-1);}
 if ((s.charAt(0) == quote1) && (s.charAt(sEnd-1) != quote1))
  {hrefString = s.substring(1,sEnd);}
 if ((s.charAt(0) != quote1) && (s.charAt(sEnd-1) != quote1))
  {hrefString = s.substring(0,sEnd);}
 /* href is relative                                             */
 if (hrefString.indexOf(":") == -1)
  {urlToString = sourceURL.toString();
   /* if href is relative to host                                */
   if (hrefString.startsWith("/"))
    {hrefString = sourceURL.getProtocol()+"://"+sourceURL.getHost()+hrefString;}
        /* if url has no "/" at end, it will be appended       */
   else {if (urlToString.endsWith("/"))
          {hrefString = urlToString+hrefString;}
        else {/* from back, the filename of sourceURL will be  */
              /* cut.                                            */
              int slashPos = urlToString.lastIndexOf("/");
              hrefString = urlToString.substring(0,slashPos+1)+hrefString;
             }
       }
  }
 hrefString = hrefString.trim().replace(quote1, quote2);
 /* String, which comes after <a href=...> up to </a>           */
 length = destURLString.length();
 lastString = destURLString.substring(end+1,length-4).trim();
 /* double quotations are replaced by single quotations         */
 /* in source-string; if they don`t get replaced you will have  */
 /* problems with the meta-symbol for double quatations.        */
 lastString = lastString.replace(quote1, quote2);
 newURLString = "<a href='"+hrefString+"'>"+lastString;
 System.out.println(newURLString);
 Link link = new Link(lastString, hrefString);
 return link;
}
public void executeDepthFirst()
```

```
  {/* Generates a hierarchy of links from this.link.dest          */
   /* with this.depth                                              */
   URL url;
   /* if URL is not loadable (e.g. error name or connection not   */
   /* possible, this method with this url will no longer be       */
   /* executed                                                     */
   try
    {url = new URL(link.dest);}
    catch (IOException e) {return;}

   String documentString = getDocumentString(url);

   /* StringTokenizer gets all hyptertext references (href) with  */
   /* method nextToken(beginString, endString, boolean)           */
   StringTokenizer strTokenizer = new StringTokenizer(documentString);

   String destURLString = strTokenizer.nextToken("<a href=", "</a>", true);
   while (! destURLString.equals(""))
    {Link link = getLink(url, destURLString);
     int newdepth = depth - 1;
     HTMLList next = new HTMLList(type, link, newdepth);
     /* if search depth is not on leafs, executes depth first     */
     if (depth > 1)
      {next.executeDepthFirst();}
     /* add the new HTMLList; if a HTMLList with equal link is in */
     /* result, it will not be inserted (look at the semantics of */
     /* addObject)                                                 */
     addObject(next);
     destURLString = strTokenizer.nextToken("<a href=", "</a>", true);
   }
  }
  public HTMLList allLeafs()
  /* generates a set of leafs (HTMLList) from a hierarchy of this */
  /* HTMLList                                                      */
  {HTMLList result = new HTMLList("ul", link, 1);
   int max = size() - 1;
   Enumeration elements = elements();
   for (int i = 0 ; i <= max ; i++)
    {Object elem = elements.nextElement();
     HTMLList ul = (HTMLList) elem;
     if (ul.size() == 0)
      {result.addObject(ul);}
     if (ul.size() > 0)
       {result.union(ul.allLeafs());}
    }
   return result;
   }
}

class VarList extends Vector
/* VarList is a Vector (ordered list of Variable). It is used to */
/* hold variables for lambda variables and all variables of the  */
/* query expression.                                              */
{public VarList()
 {super();}
 public void sort()
 {/* takes an element from unsorted this and sorts it into this. */
  /* Caution: removes double entries.                             */
  VarList thisCopy = (VarList)this.clone();
  removeAllElements();
  /* init: add first element into this                            */
  addElement(thisCopy.elementAt(0));
  int max = thisCopy.size() - 1;
  /* get all elements from thisCopy-VarList                       */
  for (int i = 1; i <= max; i++)
   {Variable elem = (Variable)thisCopy.elementAt(i);
    int maxResult = size() - 1;
    for (int j = 0; j <= maxResult; j++)
     {Variable elemResult = (Variable)elementAt(j);
      /* double Entries:                                          */
      /* if elem.name is lexically equal to elemResult.name       */
      /* doesn`t do anything: removes this entry.                 */
      if (elem.name.compareTo(elemResult.name) == 0)
        {break;}
      /* if elem is lexically smaller than elemResult             */
      if (elem.name.compareTo(elemResult.name) < 0)
        {insertElementAt(elem, j);
         break;
        }
      /* if elem is lexically the last Element                    */
      if (j == maxResult)
        {addElement(elem);
         break;
        }
     }
   }
 }
```

```
 public Variable getVar(String v)
 /* Delivers the variable named v.                              */
 {int max = size() - 1;
  for (int i = 0; i <= max; i++)
   {Variable var = (Variable)elementAt(i);
    if (var.name.equals(v))
     {return var;}
   }
  Variable va = new Variable("", new HTMLList("ul"));
  return va;
 }
}

class Link extends Object
{/* Link is an Object to represent an HTML link. Instance        */
 /* variable "source" names the source region of a link and      */
 /* "dest" names the destination URL string.                     */
 public String source;
 public String dest;

 public Link(String source, String dest)
 {this.source = source;
  this.dest = dest;
 }
 public String toHTMLString()
 /* Delivers the HTML coded string of this link.                 */
 {String result = "<a href='" + this.dest + "'>" + source + "</a>";
  return result;
 }
}

class Predicate extends Object
/* Instance variable "name" names the name of the predicate (e.g.*/
/* "Conn" or "Desc"), "arg1", "arg2" and "arg3" name the         */
/* arguments (an HTML string, a variable or a search depth).     */
/* The predivates' semantics is quasi its execution (method      */
/* "execute()").                                                 */
/* A predicate has two ("Conn") or three ("Desc") arguments,     */
/* "resultType" has value "set" or "hierarchy". A predicate with */
/* name "Desc" has resultType "hierarchy" and Predicate "Conn"   */
/* has resultType "list" as default.                             */
{public String name;
 public String arg1;
 public String arg2;
 public int arg3;
 public String resultType;
 public Responder responder;

 public Predicate(String name, String arg1, String arg2, int arg3, String resultType, Respon-
der responder)
 {this.name = name;
  this.arg1 = arg1;
  this.arg2 = arg2;
  this.arg3 = arg3;
  this.resultType = resultType;
  this.responder = responder;
 }
 public boolean isConn()
 {if (name.equals("Conn"))
    return true;
  else return false;
 }
 public boolean isDesc()
 {if (name.equals("Desc"))
    return true;
  else return false;
 }
 public boolean isBound(String var)
 /* A variable is bound, if it has a value.                      */
 {HTMLList binding = getVarValue(var);
  /* no variable                                                 */
  if (binding == null)
   {return false;}
  if (! binding.isEmpty())
   {return true;}
  return false;
 }
 public boolean isExecutable()
 /* A predicate can be executed if argument 2 is not bound and   */
 /* argument 1 is bound or argument1 is not variable (e.g. URL-  */
 /* string.                                                      */
 {if ((! isVar(arg1) || isBound(arg1)) && (! isBound(arg2)))
   {return true;}
  return false;
 }
 public boolean hasVar(String var)
 {if (arg1.equals(var) || arg2.equals(var))
```

```
   {return true;}
  return false;
 }
 public boolean hasVars()
 {if (isVar(arg1) || isVar(arg2))
     {return true;}
  return false;
 }
 public boolean isVar(String s)
 /* Variable x, y, z are for the user, Letter g is for test      */
 /* binding in logical and (boundExecute)                        */
 {return ("xyzg".indexOf(s) != -1);}
 public String equalVar(Predicate pred)
 {if (isVar(arg1) && (pred.arg1.equals(arg1) || pred.arg2.equals(arg1)))
    {return arg1;}
  if (isVar(arg2) && (pred.arg1.equals(arg2) || pred.arg2.equals(arg2)))
    {return arg2;}
  return "";
 }
 public HTMLList getVarValue(String var)
 {return (HTMLList)responder.vars.getVar(var).value;}
 public void execute()
 {/* if the first argument is an url-string and the second       */
  /* argument is a variable, binds the second variable with the  */
  /* execution of "executeDepthFirst()"                          */

  /* if both arguments are URL's, doesn`t do anything.           */
  if (! isVar(arg1) && ! isVar(arg2))
   {return;}
  if (isVar(arg2))
   {Link link = new Link("{}{}ROOT{}{}", arg1);
    HTMLList ul = new HTMLList("ul", link, arg3);
    ul.executeDepthFirst();
    if (resultType.equals("list"))
     {ul = ul.toSet();}
    bindVar(arg2, ul);
   }
 }
 public void boundExecute()
 {/* Executes if arg1 is bound. Binds the second variable. Takes */
  /* an element of bound variable and tries to get a result      */
  /* (testBinding). If testBinding is empty, the element is      */
  /* deleted from bound  variable, otherwise all elements of     */
  /* testBinding are added to second variable binding.           */
  if (isBound(arg1))
   {HTMLList firstBinding = getVarValue(arg1);
    Link link = new Link("{}{}ROOT{}{}", "bla");
    HTMLList secondBinding = new HTMLList("ul", link, 1);
    HTMLList removeList = new HTMLList("ul", link, 1);
    for (int i = 0; i <= firstBinding.size() - 1; i++)
     {Object elem = firstBinding.elementAt(i);
      HTMLList elemUL = (HTMLList) elem;
       Predicate testPred = new Predicate (name, elemUL.link.dest, "g", arg3, resultType, re-
sponder);
      testPred.execute();
      HTMLList testBinding = getVarValue("g");
      if (testBinding.isEmpty())
       {removeList.addElement(elem);}
      else
       {for (int j = 0; j <= testBinding.size() - 1; j++)
         {Object e = testBinding.elementAt(j);
          HTMLList ul = (HTMLList) e;
          secondBinding.addObject(ul);
         }
       }
     }
    firstBinding.diff(removeList);
    bindVar(arg2, secondBinding);
   }
 }
 public void and(Predicate pred)
 /* "Logical and" of this predicate and pred. Only those links   */
 /* which are loadable and have neighbour connections are        */
 /* treated (intersection of sets).                              */
 /* If both arguments of this predicate or of pred are variables */
 /* and are equal, the variable is bound to an empty Set (Error).*/
 {if ((isVar(arg1) && isVar(arg2) && arg1.equals(arg2)) ||
      (isVar(pred.arg1) && isVar(pred.arg2) && pred.arg1.equals(pred.arg2)))
   {bindVar(arg1, new HTMLList("ul"));
    return;
   }
  resultType = "list"; pred.resultType = "list";
  if (isExecutable())
    {execute();
     pred.boundExecute();
    }
  else
```

```
   {if (pred.isExecutable())
    {pred.execute();
     boundExecute();
    }
   }
 }
 public void or(Predicate pred)
 /* "Logical or" of this predicate and pred. If variables are    */
 /* equal, a union is computed else both variables are bound by  */
 /* "execute()".                                                 */
 {if (arg2.equals(pred.arg2))
   {resultType = "list"; pred.resultType = "list";
    execute();
    HTMLList firstBinding = getVarValue(arg2);
    pred.execute();
    HTMLList secondBinding = getVarValue(arg2);
    HTMLList union = firstBinding.union(secondBinding);
    bindVar(arg2, union);
   }
  else
   {execute();
    pred.execute();
   }
 }
 public void bindVar(String v, HTMLList value)
 /* Binds a value to variable v.                                 */
 {Variable var = responder.vars.getVar(v);
  var.value = value;
 }
}

class Variable extends Object
/* A variable has a name and a value.                            */
{public String name;
 public HTMLList value;

 public Variable(String name, HTMLList value)
 {this.name = name;
  this.value = value;
 }
 public boolean isEmpty()
 {if(name.equals("") || value.isEmpty())
   {return true;}
  else
   {return false;}
 }
}
```

Die Datei: Client.java

```
import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.*;
/* to presentate the HTML-Result in a javascript window          */
import netscape.javascript.JSObject;
import netscape.javascript.JSException;


/* ---------- URL Echo Sounder (URLES) - Client ---------------  */
/* URLES-Client is an Internet user interface for an easy        */
/* building of queries as expressions of the predicate lambda    */
/* calculus and for showing their answers.                       */
/* The user having loaded a client (an applet) with a standard   */
/* Internet- browser on his or her user host he or she           */
/* specifies his or her logical query mostly with mouse          */
/* operations. The user can add, remove and rename parts of the  */
/* query. Syntactically error builded inputs are showed          */
/* in seperated Error windows. The user starts his query by      */
/* pressing the "Start searching" button. For reason of security */
/* applets only build up a socket connection to a process which  */
/* is on the same host (server host) as the host which the applet*/
/* is loaded from (applet host). After this the applet           */
/* initializes (method "initSocket()") a TCP/IP based socket     */
/* connection to the URLES-Server (here with port 4712). The     */
/* applet starts a process on the user host (class               */
/* "StreamListener") which is waiting for an answer of the       */
/* URLES-Server over the sockets' input stream.                  */
/* URLES-Server starts a process on the server host with this    */
/* socket connection. The applet translates the graphical user   */
/* query into a prefix notation which is sent through the socket */
/* connection to the URLES-Server. Having executed the query the */
/* URLES-Server sends the HTML coded answer back to the waiting  */
/* applet process through the input stream of the socket         */
/* connection. The applet receives the answer and shows (here    */
/* with the Javascript understanding browser Netscape) it in its */
```

```
/* main window.                                                */

/* URLES-Client has the following properties (see URLES-Server   */
/* for detailed information):                                  */
/* 1. it is blocked up till the server sends the answer.       */
/* 2. it is reliable.                                          */
/* 3. it is buffered.                                          */
/* 4. it uses internet addresses.                              */
/* 5. it is open to security.                                  */

/* At the moment URLES-Client is limited to the predicates       */
/* "Conn" and "Desc", to search depth "4", to a maximum of 2     */
/* logical expressions, and to the logical operators "and" and   */
/* "or", backward connectionsare not possible and cycles will not*/
/* be seen.                                                    */

public class URLESClient extends Applet
{public static final int PORT = 4712;
 Socket s;
 DataInputStream in;
 PrintStream out;
 StreamListener listener;
 Frame parent;
 Label  title, label3, labelOpenBracket1, labelOpenBracket2, labelCloseBracket1,
        labelCloseBracket2, labelComma1, labelComma2, labelComma3, labelComma4,
        result;
 Button addExprButton, removeExprButton, searchButton, finishSearchButton;
 Choice predChoice1, termChoice1, termChoice2, depthChoice1,
        varChoice1, varChoice2, varChoice3, varChoice4, logOpChoice,
        predChoice2, termChoice3, termChoice4, depthChoice2, lambdaChoice1,
        lambdaChoice2 ;
 Panel  expressionPanel1, logOpPanel, expressionPanel2;
 Font   fo20, fo30;
 TextField inputFieldURL1, inputFieldURL2, inputFieldURL3, inputFieldURL4;

 public void init()
 /* Initializes the applet window with user adequate labels,     */
 /* buttons, textFields and choice menus for manipulating the    */
 /* logical query.                                             */
 {setLayout(new BorderLayout());
  setFont(new Font("TimesRoman", Font.PLAIN, 12));

  title = new Label("URL Echo Sounder (URLES)");
  fo20 = new Font("TimesRoman", Font.BOLD, 20);
  title.setFont(fo20);
  title.reshape(10,10,400,30);
  add(title);

  /* Add log Expression                                        */
  addExprButton = new Button("Add Expression");
  addExprButton.reshape(70,50,90,30);
  add(addExprButton);

  /* Remove log Expression                                     */
  removeExprButton = new Button("Remove Expression");
  removeExprButton.reshape(200,50,110,30);
  add(removeExprButton);

  /* Query Expression Panel1 with FlowLayout                   */
  expressionPanel1 = new Panel();
  expressionPanel1.setLayout(new FlowLayout(FlowLayout.LEFT));
  expressionPanel1.reshape(10,90,600,60);

  /* Predicate LogOpChoice                                     */
  logOpChoice = new Choice();
  logOpChoice.addItem("And");
  logOpChoice.addItem("Or");

  /* Log Operator Panel                                        */
  logOpPanel = new Panel();
  logOpPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
  logOpPanel.add(logOpChoice);
  logOpPanel.reshape(50,150,80,30);

  /* Query Expression Panel2 with FlowLayout                   */
  expressionPanel2 = new Panel();
  expressionPanel2.setLayout(new FlowLayout(FlowLayout.LEFT));
  expressionPanel2.reshape(10,180,600,60);

  /* Predicate Choice 1                                        */
  predChoice1 = new Choice();
  predChoice1.addItem("Conn");
  predChoice1.addItem("Desc");

  /* Open Bracket Label 1                                      */
  labelOpenBracket1 = new Label ("(");
  fo30 = new Font("TimesRoman", Font.BOLD, 30);
```

```
labelOpenBracket1.setFont(fo30);

/* Term Choice 1                                               */
termChoice1 = new Choice();
termChoice1.addItem("URL");
termChoice1.addItem("Var");

inputFieldURL1 = new TextField(50);
inputFieldURL1.setText("http://anaconda.cs.tu-berlin.de:1111/test/index.html");

inputFieldURL2 = new TextField(50);
inputFieldURL2.setText("http://anaconda.cs.tu-berlin.de:1111/test/");

inputFieldURL3 = new TextField(50);
inputFieldURL3.setText("http://anaconda.cs.tu-berlin.de:1111/test/");

inputFieldURL4 = new TextField(50);
inputFieldURL4.setText("http://anaconda.cs.tu-berlin.de:1111/test/");

/* comma Label1                                                */
labelComma1 = new Label (",");
labelComma1.setFont(fo30);

/* Term Choice2                                                */
termChoice2 = new Choice();
termChoice2.addItem("URL");
termChoice2.addItem("Var");

/* Var Choice 1                                                */
varChoice1 = new Choice();
varChoice1.addItem("x");
varChoice1.addItem("y");
varChoice1.addItem("z");

/* Var Choice 2                                                */
varChoice2 = new Choice();
varChoice2.addItem("x");
varChoice2.addItem("y");
varChoice2.addItem("z");

/* Var Choice 3                                                */
varChoice3 = new Choice();
varChoice3.addItem("x");
varChoice3.addItem("y");
varChoice3.addItem("z");

/* Var Choice 4                                                */
varChoice4 = new Choice();
varChoice4.addItem("x");
varChoice4.addItem("y");
varChoice4.addItem("z");

/* comma Label2                                                */
labelComma2 = new Label (",");
labelComma2.setFont(fo30);

/* Depth Choice 1                                              */
depthChoice1 = new Choice();
depthChoice1.addItem("1");
depthChoice1.addItem("2");
depthChoice1.addItem("3");
depthChoice1.addItem("4");

/* Close Bracket Label 1                                       */
labelCloseBracket1 = new Label (")");
labelCloseBracket1.setFont(fo30);

/* Predicate Choice 2                                          */
predChoice2 = new Choice();
predChoice2.addItem("Conn");
predChoice2.addItem("Desc");

/* Open Bracket Label 2                                        */
labelOpenBracket2 = new Label ("(");
fo30 = new Font("TimesRoman", Font.BOLD, 30);
labelOpenBracket2.setFont(fo30);

/* Term Choice 3                                               */
termChoice3 = new Choice();
termChoice3.addItem("URL");
termChoice3.addItem("Var");

/* comma Label3                                                */
labelComma3 = new Label (",");
labelComma3.setFont(fo30);

/* Term Choice4                                                */
```

```
  termChoice4 = new Choice();
  termChoice4.addItem("URL");
  termChoice4.addItem("Var");

  /* comma Label4                                                */
  labelComma4 = new Label (",");
  labelComma4.setFont(fo30);

  /* Depth Choice 2                                              */
  depthChoice2 = new Choice();
  depthChoice2.addItem("1");
  depthChoice2.addItem("2");
  depthChoice2.addItem("3");
  depthChoice2.addItem("4");

  /* Close Bracket Label 2                                       */
  labelCloseBracket2 = new Label (")");
  labelCloseBracket2.setFont(fo30);

  /* Query-Result Label                                          */
  result = new Label("Query-Result: ");
  fo20 = new Font("TimesRoman", Font.BOLD, 20);
  result.setFont(fo20);
  result.reshape(10,320,140,30);
  add(result);

  /* Lambda Choice 1                                             */
  lambdaChoice1 = new Choice();
  lambdaChoice1.addItem("None");
  lambdaChoice1.addItem('\u03BB'+"x");
  lambdaChoice1.addItem('\u03BB'+"y");
  lambdaChoice1.addItem('\u03BB'+"z");
  lambdaChoice1.select(1);
  lambdaChoice1.reshape(160, 320, 60, 30);
  add(lambdaChoice1);

  /* Lambda Choice 2                                             */
  lambdaChoice2 = new Choice();
  lambdaChoice2.addItem("None");
  lambdaChoice2.addItem('\u03BB'+"x");
  lambdaChoice2.addItem('\u03BB'+"y");
  lambdaChoice2.addItem('\u03BB'+"z");
  lambdaChoice2.reshape(240, 320, 60, 30);
  add(lambdaChoice2);

  /* Start Search Button                                         */
  searchButton = new Button("Start searching");
  searchButton.reshape(70,400,100,30);
  add(searchButton);

  /* Stop Search Button                                          */
  finishSearchButton = new Button("Stop searching");
  finishSearchButton.reshape(200,400,100,30);
  add(finishSearchButton);

  initSocket();

 }
public void initSocket()                                                  FLAN
/* Initializes a socket to communicate with URLES-Server on port */
/* 4712. Creates input and output streams for the socket.        */
/* use with the socket. Creates a thread to wait for the answer. */
{try                                                                      FLAN
  {if (s != null)                                                         FLAN
    {s.close();}                                                          FLAN
   try                                                                    FLAN
    {s = new Socket(getCodeBase().getHost(), PORT);                       FLAN
     in = new DataInputStream(s.getInputStream());                        FLAN
     out = new PrintStream(s.getOutputStream());                          FLAN
     listener = new StreamListener(in, this);                             FLAN
    }                                                                     FLAN
   catch (IOException e) {showStatus(e.toString());}                      FLAN
  }                                                                       FLAN
  catch (IOException e) {showStatus(e.toString());}                       FLAN

  /* set parent frame                                            */
  Object frame = getParent();
  while (! (frame instanceof Frame))
    frame = ((Component) frame).getParent();
  parent = (Frame)frame;

  /* set Cursor back to standard arrow for the applet            */
  parent.setCursor(Frame.DEFAULT_CURSOR);
}
public boolean action(Event e, Object what)
/* When the user fills out the forms, sends it to the server.    */
{if (e.target == predChoice1)
```

```
{/*  predicate choice   */
 if (predChoice1.getSelectedItem() == "Conn")
  {expressionPanel1.remove(labelComma2);
   expressionPanel1.remove(depthChoice1);
   expressionPanel1.layout();
   expressionPanel1.repaint();
  }
 if (predChoice1.getSelectedItem() == "Desc")
  {expressionPanel1.add(labelComma2, 5);
   depthChoice1.select("1");
   expressionPanel1.add(depthChoice1, 6);
   expressionPanel1.layout();
   expressionPanel1.repaint();
  }
 }
 if (e.target == predChoice2)
  {/*  predicate choice   */
   if (predChoice2.getSelectedItem() == "Conn")
    {expressionPanel2.remove(labelComma4);
     expressionPanel2.remove(depthChoice2);
     expressionPanel2.layout();
     expressionPanel2.repaint();
    }
   if (predChoice2.getSelectedItem() == "Desc")
    {expressionPanel2.add(labelComma4, 5);
     depthChoice2.select("1");
     expressionPanel2.add(depthChoice2, 6);
     expressionPanel2.layout();
     expressionPanel2.repaint();
    }
   }
  if (e.target == termChoice1)
   {/*  predicate choice   */
    expressionPanel1.remove(termChoice1);
    if (termChoice1.getSelectedItem() == "URL")
     {expressionPanel1.add(inputFieldURL1, 2);}
      if (termChoice1.getSelectedItem() == "Var")
       {expressionPanel1.add(varChoice1, 2);}
      expressionPanel1.layout();
      expressionPanel1.repaint();
     }
    if (e.target == termChoice2)
     {/*  predicate choice   */
      expressionPanel1.remove(termChoice2);
      if (termChoice2.getSelectedItem() == "URL")
       {expressionPanel1.add(inputFieldURL2, 4);}
      if (termChoice2.getSelectedItem() == "Var")
       {expressionPanel1.add(varChoice2, 4);}
      expressionPanel1.layout();
      expressionPanel1.repaint();
     }
    if (e.target == termChoice3)
     {/*  predicate choice   */
      expressionPanel2.remove(termChoice3);
      if (termChoice3.getSelectedItem() == "URL")
       {expressionPanel2.add(inputFieldURL3, 2);}
      if (termChoice3.getSelectedItem() == "Var")
       {expressionPanel2.add(varChoice3, 2);}
      expressionPanel2.layout();
      expressionPanel2.repaint();
     }
    if (e.target == termChoice4)
     {/*  predicate choice   */
      expressionPanel2.remove(termChoice4);
      if (termChoice4.getSelectedItem() == "URL")
       {expressionPanel2.add(inputFieldURL4, 4);}
      if (termChoice4.getSelectedItem() == "Var")
       {expressionPanel2.add(varChoice4, 4);}
      expressionPanel2.layout();
      expressionPanel2.repaint();
     }
    if (e.target == addExprButton)
     {if (expressionPanel1.getComponents().length == 0)
       {predChoice1.select("Conn");
        termChoice1.select("URL");
        termChoice2.select("URL");
        depthChoice1.select("1");
        expressionPanel1.add(predChoice1);
        expressionPanel1.add(labelOpenBracket1);
        expressionPanel1.add(termChoice1);
        expressionPanel1.add(labelComma1);
        expressionPanel1.add(termChoice2);
        expressionPanel1.add(labelCloseBracket1);
        add(expressionPanel1);
        expressionPanel1.layout();
        expressionPanel1.repaint();
        layout();
```

```
      repaint();
     }
    else
     {if (expressionPanel2.getComponents().length == 0)
       {predChoice2.select("Conn");
        logOpChoice.select("And");
        termChoice3.select("URL");
        termChoice4.select("URL");
        depthChoice2.select("1");
        expressionPanel2.add(predChoice2);
        expressionPanel2.add(labelOpenBracket2);
        expressionPanel2.add(termChoice3);
        expressionPanel2.add(labelComma3);
        expressionPanel2.add(termChoice4);
        expressionPanel2.add(labelCloseBracket2);
        add(expressionPanel2);
        add(logOpPanel);
        logOpPanel.layout();
        logOpPanel.repaint();
        expressionPanel2.layout();
        expressionPanel2.repaint();
        layout();
        repaint();
       }
     }
   }
  if (e.target == removeExprButton)
   {if (expressionPanel2.getComponents().length == 0)
     {expressionPanel1.removeAll();
      remove(expressionPanel1);
      layout();
      repaint();
     }
    else
     {expressionPanel2.removeAll();
      remove(logOpPanel);
      remove(expressionPanel2);
      layout();
      repaint();
     }
   }
  if (e.target == searchButton)
   {Component[] components = expressionPanel1.getComponents();
    Component[] comps = expressionPanel2.getComponents();
    String lambdaVar1 = lambdaChoice1.getSelectedItem();
    if (lambdaVar1.equals("None"))
     {lambdaVar1 = "";}
    else {lambdaVar1 = lambdaVar1.substring(1);}
    String lambdaVar2 = lambdaChoice2.getSelectedItem();
    if (lambdaVar2.equals("None"))
     {lambdaVar2 = "";}
    else {lambdaVar2 = lambdaVar2.substring(1);}
    String var1 = varChoice1.getSelectedItem();
    String var2 = varChoice2.getSelectedItem();
    String var3 = varChoice3.getSelectedItem();
    String var4 = varChoice4.getSelectedItem();
    String depth1 = depthChoice1.getSelectedItem();
    String depth2 = depthChoice2.getSelectedItem();
    String pred1 = predChoice1.getSelectedItem();
    String pred2 = predChoice2.getSelectedItem();
    String logOp = logOpChoice.getSelectedItem();
    String url1 = inputFieldURL1.getText();
    String url2 = inputFieldURL2.getText();
    String url3 = inputFieldURL3.getText();
    String url4 = inputFieldURL4.getText();

    if (components.length == 0)
     {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
      errDialog.showError("No query specified.");
      return true;
     }
    /* Arguments in first predicate are not yet specified    */
    if (components[2] == termChoice1 || components[4] == termChoice2)
     {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
      errDialog.showError("Argument(s) in first predicate is (are) \nnot specified.");
      return true;
     }
    if (components[2] == inputFieldURL1 && components[4] == inputFieldURL2)
     {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
      errDialog.showError("Arguments in first predicate are both URL");
      return true;
     }
    if (lambdaVar1.equals("") && lambdaVar2.equals(""))
     {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
      errDialog.showError("Lambda-Variable is not specified.");
      return true;
     }
```

```
        /*  query without bool operator                         */
        if (comps.length == 0)
         {/* 1. argument of predicate is variable               */
          if (components[2] == varChoice1)
           {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
             errDialog.showError("The determination of the answer would spent \ntoo much time
(left argument should not be Variable)");
            return true;
           }
          /* first attribute: URL ; second attribute: Variable   */
          if (components[2] == inputFieldURL1 && components[4] == varChoice2)
           {/* Lambda variable 1 not bound                       */
            if (lambdaVar1.equals(var2) || lambdaVar1.equals(""))
              {;}
            else
             {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
              errDialog.showError("Lambda-Variable: "+lambdaVar1+" is not bound.");
              return true;
             }
            /* Lambda variable 2 not bound                       */
            if (lambdaVar2.equals(var2) || lambdaVar2.equals(""))
             {;}
            else
             {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
              errDialog.showError("Lambda-Variable: "+lambdaVar2+" is not bound.");
              return true;
             }
            }
          /* Cursor is set to "Waiting" (clock)                  */
          parent.setCursor(Frame.WAIT_CURSOR);
          /* simple Desc-predicate   */
          if (predChoice1.getSelectedItem() == "Desc")
           {/*Query is sent to HyQL-Server in prefix-notation */
              out.println("Desc"+" "+url1+" "+var2+" "+depth1+" "+lambdaVar1+" "+lambdaVar2+"
");
           }
          /* simple Conn-predicate                               */
          else
           {out.println("Conn"+" "+url1+" "+var2+" "+lambdaVar1+" "+lambdaVar2+" ");}
          return true;
         }
        /*  query with bool operator                            */
        else
         {/* Arguments in second predicate are not yet specified */
          if (comps[2] == termChoice3 || comps[4] == termChoice4)
           {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
            errDialog.showError("Argument(s) in second predicate is (are) \nnot specified.");
            return true;
           }
          if (comps[2] == inputFieldURL3 && comps[4] == inputFieldURL4)
           {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
            errDialog.showError("Arguments in second predicate are both URL.");
            return true;
           }
          /* Lambda variable 1 is not bound                      */
           if (lambdaVar1.equals(var2) || lambdaVar1.equals(var3) || lambdaVar1.equals(var4) ||
lambdaVar1.equals(""))
           {;}
          else
           {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
            errDialog.showError("Lambda-Variable: "+lambdaVar1+" is not bound.");
            return true;
           }
          /* Lambda variable 2 is not bound                      */
           if (lambdaVar2.equals(var2) || lambdaVar2.equals(var3) || lambdaVar2.equals(var4) ||
lambdaVar2.equals(""))
           {;}
          else
           {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
            errDialog.showError("Lambda-Variable: "+lambdaVar2+" is not bound.");
            return true;
           }
          if (logOp.equals("Or"))
           {if (components[2] == varChoice1 || comps[2] == varChoice3)
             {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
              errDialog.showError("The determination of the answer would spent \ntoo much time
(left argument should not be Variable)");
              return true;
             }
           }
          if (logOp.equals("And"))
           {/* not: (2. argument of 1. predicate and 1. argument */
            /*       of 2. predicate are variables and are equal)*/
            /* or:                                               */
            /* not: (1. argument of 1. predicate and 2. argument */
            /*       of 2. predicate are variables and are equal)*/
```

```
            if (!((components[4] == varChoice2 && comps[2] == varChoice3 && var2.equals(var3))
||
                  (components[2] == varChoice1 && comps[4] == varChoice4 && var1.equals(var4))
))
             {ErrorDialog errDialog = new ErrorDialog(parent, "Error", true);
               errDialog.showError("Variables in 2. argument of 1. predicate and \n1. argument
of 2. predicate of your logical AND Expression are \nnot equal. \n or:\nVariables in 1. argu-
ment of 1. predicate and \n2. argument of 2. predicate of your logical AND Expression are
\nnot equal.");
              return true;
             }
           }

          parent.setCursor(Frame.WAIT_CURSOR);
          String infixQueryString = "";
          infixQueryString = infixQueryString + logOp + " ";
          /* first predicate    */
          if (pred1.equals("Conn"))
           {infixQueryString = infixQueryString + pred1 + " ";
            if (components[2] == varChoice1)
             {infixQueryString = infixQueryString + var1 + " ";}
            else {infixQueryString = infixQueryString + url1 + " ";}
            if (components[4] == varChoice2)
             {infixQueryString = infixQueryString + var2 + " ";}
            else {infixQueryString = infixQueryString + url2 + " ";}
           }
          if (pred1.equals("Desc"))
           {infixQueryString = infixQueryString + pred1 + " ";
            if (components[2] == varChoice1)
             {infixQueryString = infixQueryString + var1 + " ";}
            else {infixQueryString = infixQueryString + url1 + " ";}
            if (components[4] == varChoice2)
             {infixQueryString = infixQueryString + var2 + " ";}
            else {infixQueryString = infixQueryString + url2 + " ";}
            infixQueryString = infixQueryString + depth1 + " ";
           }
          /* second predicate                                    */
          if (pred2.equals("Conn"))
           {infixQueryString = infixQueryString + pred2 + " ";
            if (comps[2] == varChoice3)
             {infixQueryString = infixQueryString + var3 + " ";}
            else {infixQueryString = infixQueryString + url3 + " ";}
            if (comps[4] == varChoice4)
             {infixQueryString = infixQueryString + var4 + " ";}
            else {infixQueryString = infixQueryString + url4 + " ";}
           }
          if (pred2.equals("Desc"))
           {infixQueryString = infixQueryString + pred2 + " ";
            if (comps[2] == varChoice3)
             {infixQueryString = infixQueryString + var3 + " ";}
            else {infixQueryString = infixQueryString + url3 + " ";}
            if (comps[4] == varChoice4)
             {infixQueryString = infixQueryString + var4 + " ";}
            else {infixQueryString = infixQueryString + url4 + " ";}
            infixQueryString = infixQueryString + depth2 + " ";
           }
          infixQueryString = infixQueryString + lambdaVar1 + " " +lambdaVar2 + " ";
          out.println(infixQueryString);
         return true;
         }
       }
      if (e.target == finishSearchButton)
       {initSocket();}
      return false;
    }
}

class ErrorDialog extends Dialog
{/* Simple error window with label                               */
private TextArea errorArea;

public ErrorDialog(Frame parent, String name, boolean modal)
{super (parent, name, modal);
 resize(400,150);
 add("South", new Button("ok"));
 errorArea = new TextArea(4,10);
 errorArea.setEditable(false);
 add("Center", errorArea);
}
public void showError(String error)
{errorArea.appendText(error);
 show();
}
public boolean action(Event evt, Object what)
{if ("ok".equals(what))
   dispose();
 return true;
```

```
}
}

class StreamListener extends Thread
/* Waits for input from the URLES-Server on the specified input  */
/* stream. When the input is sent, it will be shown in the       */
/* internet browsers main window (here: a Javascript             */
/* understanding browser).                                       */
{DataInputStream in;
 URLESClient urlesClient;

public StreamListener(DataInputStream input, URLESClient cl)
{in = input;
 urlesClient = cl;
 start();
}
public void showData(String data)
{/* set cursor back to normal arrow                              */
 Object frame = urlesClient.getParent();
 while (! (frame instanceof Frame))
  frame = ((Component) frame).getParent();
 ((Frame)frame).setCursor(Frame.DEFAULT_CURSOR);
 /*  show result                                                 */
 JSObject win = JSObject.getWindow(urlesClient);
 JSObject doc = (JSObject) win.getMember("document");
 urlesClient.initSocket();
 doc.eval("writeln(\""+data+"\"); close();");
}
public void run()
{String line;
 String htmlDocument = "";
 for(;;)
  {try
    {line = in.readLine();
     /* answer is read line by line till empty line              */
     if (line == null)
       {this.showData(htmlDocument);}
     htmlDocument = htmlDocument + line;
    }
   catch (IOException e1){System.err.println("Error: "+e1);}
  }
 }
}
```

Die Datei: index.html

```
<html><head>
<title>HyQL-Client</title>
</head>
<body>
<applet   codebase="http://anaconda.cs.tu-berlin.de:1111/applets/URLESClient/URLESClient.prj/"
code="URLESClient.class" width=600 height=600 MAYSCRIPT></applet>


</body></head>
```

Die Methode: nextToken

```
package java.util;

import java.lang.*;

public String nextToken(String beginString, String endString, boolean ignoreCase)
/* Delivers next token which begins with "beginString" */
/* and ends with "endString".                          */
/* "ignoreCase" is a flag for swithing case on         */
/* ("false") or off ("true").                          */
{int endStringLength = endString.length();
 int start = str.indexOf(beginString, currentPosition, true);
 int end = str.indexOf(endString, start, true);
 if ((start == -1) || (end == -1))
   {currentPosition = maxPosition; return ""; }
 currentPosition = end+endStringLength;
 return str.substring(start,currentPosition);
}
```

Erklärung

Hiermit versichere ich, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Diese Arbeit ist bisher noch nicht anderweitig als Dissertation eingereicht oder veröffentlicht worden.

Berlin im Juni 2001

Josef Willenborg
Digitally signed by Josef Willenborg
DN: cn=Josef Willenborg
Date: 2001.09.15 15:21:14 Z
Signature Not Verified

(Josef Willenborg)